DAVIC

Digital Audio-Visual Council

DAVIC 1.5 Specifications

Revision 6.0

TV Anytime and *TV Anywhere*

Editors:

Henry Chadwick IBM e-mail: hdchad@us.ibm.com
Masahisa Kawashima NTT e-mail: Kawashima.Masahisa@nslab.ntt.co.jp
Jan van der Meer Philips Consumer Electronics e-mail: jan.vandermeer@ehv.ce.philips.com
Jon Piesing, Ronald Tol e-mails : *jon@prl.research.philips.com*, *tolr@natlab.research.philips.com*

TV Anytime and *TV Anywhere*

Published by Digital Audio-Visual Council

Geneva, Switzerland

Table of Contents

PREFACE **1**

1 REFERENCES AND DEFINITIONS **2**

1.1 NORMATIVE REFERENCES 2
1.2 INFORMATIVEREFERENCES 2

2 ACRONYMS AND ABBREVIATIONS **3**

3 TV ANYTIME AND TV ANYWHERE -APPLICATION SCENARIOS AND REQUIREMENTS- **6**

3.1 INTRODUCTION 6
3.2 KEY CONCEPTS 7
3.2.1 Source content is located in time and space *7*
3.2.2 Selection *8*
3.2.3 Fulfillment *8*
3.2.4 Management and Use of Acquired Content *10*
3.3 APPLICATION SCENARIOS 10
3.3.1 Broadcast Capture *11*
3.3.2 File Transfer from Remote Storage Devices *13*
3.3.3 Remote Stream Access *14*
3.3.4 Web Links *16*
3.3.5 Segment jumping *17*
3.3.6 Content Customization *17*
3.4 REQUIREMENTS 19
3.4.1 Content Fulfillment and Selection *19*
3.4.2 Content Rights and Security *21*
3.4.3 Device and Content Management *22*
3.5 IMPLEMENTATION CONSIDERATIONS 23
3.5.1 Content Usability *23*
3.5.2 Resource Use *24*

4 TV ANYTIME AND TV ANYWHERE -SYSTEM SPECIFICATION- **27**

4.1 INTRODUCTION 27
4.2 A LIST OF NORMATIVE REFERENCES 27
4.3 SYSTEM MODEL 27
4.4 TOOL DEFINITIONS AND REFERENCES 30
4.4.1 Meta-data service tool *30*
4.4.2 Location resolution tool *31*
4.4.3 Content management and protection tool *34*
4.4.4 Local storage control tool *34*
4.4.5 Media-Object Service tool *34*
4.5 WALK-THROUGH EXAMPLES FOR TYPICAL TV ANYTIME AND TV ANYWHERE APPLICATIONS 37
4.5.1 Phase 1: Selection *phase* *37*
4.5.2 Phase 2: Capture *Pre-arrangement phase* *38*
4.5.3 Phase 3: Capture *phase* *38*
4.5.4 Phase 4: Presentation *Pre-arrangement phase* *38*
4.5.5 Phase 5: Presentation *phase* *38*

5 TV ANYTIME AND TV ANYWHERE -TOOLS- 40

5.1 INFORMATION REPRESENTATION 40
5.1.1 Audio and Speech for TV Anywhere 40
5.1.2 Video for TV Anywhere 40
5.1.3 MPEG-2 broadcast programmes for TV Anytime 40
5.1.4 Program Index System 40
5.1.5 Programme Identifiers and Locators 61
5.1.6 Metadata for TV Anytime and TV Anywhere 66
5.2 JAVA APIS FOR TV ANYTIME AND TV ANYWHERE - THE "ORG.DAVIC.STORAGE" PACKAGE AND SUPPORTING CLASSES 110
5.2.1 Objective 110
5.2.2 Requirements 110
5.2.3 General 111
5.2.4 Specification of Supporting Classes 111
5.2.5 Media Playback API Specification 116
5.2.6 Recording API Specification 116

ANNEX A 126

ANNEX B 138

ANNEX C 139

ANNEX D 140

Table of Figures

Figure 1 : Source Content is Located in Time and Space 7
Figure 2 : Selection Process 8
Figure 3 : Illustration of Required Processes of Selection, Decomposition and Resolution 9
Figure 4 : Content Selection and Fulfillment Processes of Broadcast Capture 11
Figure 5 : Content Selection and Fulfillment 20
Figure 6 : System Model for DAVIC TV Anytime and TV Anywhere 29
Figure 7 : Relation between application and system model 29
Figure 8 : Composition of meta-data service tool 30
Figure 9 : Composition of location resolution tools 32
Figure 10 : local storage control tool 34
Figure 11 : Composition of media-object service tool 35
Figure 12 : Decomposition of media-object delivery tool 35
Figure 13 : Walk-through examples for typical TV Anytime and TV Anywhere applications 37
Figure 14 : Walk-through example of a typical application 39
Figure 15 : Structure of EIT 42
Figure 16 : Structure of LIT 43
Figure 17 : Structure of ERT 43
Figure 18 : ERT Tree structure 44
Figure 19 : Model of a hierarchical structured program 45
Figure 20 : Hierarchical event model mapping 45
Figure 21 : Model of Multi-scenario program 46
Figure 22 : Event table data model in the multi-scenario structure 47
Figure 23 : General Information on Program Group Index 49
Figure 24 : General Information on Program Segment Index 50
Figure 25 : Search, selection, and fulfilment mechanisms 64
Figure 26 : Example of UPI based resolving mechanism 64
Figure 27 : DAVIC Metadata system reference model 67
Figure 28 : TV-Anytime and TV-Anwhere content and metadata flows 68
Figure 29 : Example illustrating the relationship between the CP, CI, CIE and CIE Segments / Objects and associated metadata 69
Figure 30 : Metadata access restriction and filtering layers 73
Figure 31 : Metadata Coding for DAVIC Core Schema 76
Figure 32 : Metadata Package 80
Figure 33 : Layered Filtering for Preferred Locator Extraction 81
Figure 34 : All of Original Metadata is mapped onto bit pattern 82
Figure 35 : A Part of Original Metadata is mapped onto bit pattern 82
Figure 36 : Class hierarchies defined in RDF 83
Figure 37 : Mask Schema 84
Figure 38 : Filtering Mask Structure 84
Figure 39 : Metadata schema and its instance 85
Figure 40 : Metadata Schema Extension 88
Figure 41 : Metadata Package Section 90
Figure 42 : Metadata Schema Section 95
Figure 43 : Tables and descriptors for metadata filtering system 98

TABLES

Table 1 *Defined DAVIC tools and relevant sections* *30*
Table 2 *Derived tools for meta-data delivery tool* *31*
Table 3 *a list of the locator formats adopted for DAVIC TV Anytime and TV Anywhere (I)* *33*
Table 4 *Derived types of locator delivery tool* *33*
Table 5 *Derived types of streaming tool* *35*
Table 6 *Derived types of file transfer tool* *37*
Table 5-1 *Local Event Information Section* *50*
Table 5-2 *Event Relation Section* *52*
Table 5-3 *Relation Type* *53*
Table 5-4 *Characteristics of Collection* *53*
Table 5-5 *Index Transmission information Table* *54*
Table 5-6 *Basic Local Event Descriptor* *55*
Table 5-7 *Segmentation mode* *56*
Table 5-8 *Reference Descriptor* *57*
Table 5-9 *Node Relation Descriptor* *58*
Table 5-10 *Reference type* *58*
Table 5-11 *Short Node Information Descriptor* *59*
Table 5-12. *STC Reference Descriptor* *59*
Table 5-13 *STC reference mode* *60*
Table 5-14 *Allocation of the tag value and possible locations of the descriptors* *61*
Table 5-15: *Examples of "core" metadata* *70*
Table 5-16: *Examples of "version" metadata* *71*
Table 5-17: *Dublin core categorisation and classification* *72*
Table 5-18. *Syntax of Metadata Package Section* *90*
Table 5-19. *Syntax of Metadata Descriptor* *92*
Table 5-20. *Syntax of Locator Descriptor* *93*
Table 5-21. *Definition of locator_syntax_identifier* *93*
Table 5-22. *Syntax of Filtering Mask Descriptor* *94*
Table 5-23. *Syntax of Metadata Schema Descriptor* *96*

TV Anytime and *TV Anywhere*

Preface

About This Specification

This document contains the DAVIC Specification for *TV Anytime* and *TV Anywhere*. The document mainly consists of three sections:

- Section 3 – Application Scenarios and Requirements
- Section 4 – System Specficications
- Section 5 – Tools

Each section provides an additional level of detail for those who are interested in the DAVIC *TV Anytime* and *TV Anywhere* system. Those who only want to understand the function of the system and what it does can read only Section 3. Those who want a more detailed understanding of how the system operates can read Section 4, and those who want to implement the system and need to understand the specific details can read Section 5. To understand each section one must read the preceding sections first.

1 References and Definitions

1.1 Normative References

ISO/IEC 13818–1	Information technology—Generic coding of moving pictures and associated audio information—Part 1: Systems (Note: known as MPEG-2 Systems)
ISO/IEC 13818–6	Information technology—Generic coding of moving pictures and associated audio information—Part 6: Digital Storage Media Command and Control (DSM-CC)
ISO/IEC 14496	Coding of audio-visual objects (MPEG-4)
ISO 639-2	Terminology - Codes for the representation of names of languages
ISO 8601	Data elements and interchange formats – Information interchange – Representation of dates and times
ISO/IEC 8859-1	Information technology - 8-bit single-byte coded graphic character sets, Latin alphabets
ISO 10646	Information technology - Universal multiple-octet coded character set (UCS)
IETF RFC 791	Internet Protocol (IP Addressing)
IETF RFC 793	Transmission Control Protocol (TCP)
IETF RFC-1889	RTP: A Transport Protocol for Real-Time Applications
IETF RFC-1890	RTP Profile for Audio and Video Conferences with Minimal Control
IETF RFC 2068	Hypertext Transport Protocol –HTTP/1.1
IETF RFC-2205	Resource ReSerVation Protocol (RSVP) -- Version 1.0 Functional Specification
IETF RFC-2326	Real Time Streaming Protocol (RTSP)
IETF RFC-2327	SDP : Session Description Protocol
IETF RFC 2141	URN Syntax
IETF RFC 2250	RTP Payload format for MPEG1/MPEG2 Video
ETS 300 468	Specification for Service Information (SI) in DVB Systems (January 1997)
ARIB STD-B10	ARIB-SI
DAVIC	DAVIC 1.4.1 specifications
DAVIC	DAVIC Intranet Platform (I) specifications

1.2 InformativeReferences

-

2 Acronyms and abbreviations

AAC	Advanced Audio Coding
ARIB	Asociation of Radio Indusries and Businesses
ATM	Asynchronous Transfer Mode
BCD	Binary-Coded Decimal number
BIFS	Binary Format for Scene
CBR	Constant Bit Rate
CELP	Code Excited Linear Prediction
CIE	Content Item Element
CIEO	Content Item Element Object
CIES	Content Item Element Segment
CRC	Cycric Redundancy Code
CSS	Cascading Style Sheets
DC	Dublin Core
DEMUX	DE-Multiplex
DSM-CC	Digital Storage Media Command and Control
DVB	Digital Video Broadcasting
EBU	European Broadcasting Union
EIT	Event Information Table
EPG	Electronic Program Guide
ERT	Event Relation Table
FM	Frequency Modulation
G	Guide
GIF	Graphics Interchange Format
GSM	Global System Mobile
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
IEC	International Electrotechnical Commission
IP	Internet Protocol
IPMP	Intellectual Property Management and Protection
IPR	Intellectual Propaty Right
ISO	International Organization for Standardization
ITT	Index Transmission information Table
JMF	JAVA Media Framework

JPEG	Joint Picture Experts Group
LIT	Local event Information Table
MPEG	Moving Pictures Expert Group
NID	Namespace Identifier
NPT	Normal Play Time
NSS	Namespace Specific String
NTSC	National Television System Committee
NVOD	Near Video On Demand
OCI	Object Content Information
PAL	Phase Alternating Line
PG	Program Guide
PID	Packet Identifier
PMT	Program Map Table
POT	Program-index On-stream Table
PS	Program Stream
PTS	Presentation Time Stamp
RA	Resolving Authority
RDF	Resource Description Framework
RDM	Reference Decoder Model
RFC	Request For Comment
RTP	Real Time Protocol
SECAM	Sequential Colour with Memory
SI	Service Information
SMPTE	Society of Motion Picture & Televion Engineers
SNID	Sub Namespace Identifier
SNSS	Sub Namespace Specific String
STB	Set Top Box
STC	System Time Clock
STD	System Target Decoder
STU	Set Top Unit
TBD	To Be Defined
TCP	Transmission Control Protocol
TS	Transport Stream
TTS	Text To Speech
Twin VQ	Twin Vector Quantization
UPI	Universal Programme Identifier

URL	Unique Resource Locator
URN	Uniform Resource Name
UTC	Universal Time Co-ordinated
VBR	Valuable Bit Rate
VRML	Virtual Reality Modeling Language
W3C	World Wide Web Consociam
WIPO	World Intellectual Property Organisation
WWW	World Wide Web
XML	eXtensible Markup Language

3 TV Anytime and TV Anywhere -Application Scenarios and Requirements-

3.1 Introduction

The delivery of Audio-visual material in a digital form to the home creates the fundamental opportunity for consumer systems to store content on digital media, such as magnetic disks and tape or optical disk drives. This opportunity can be realized both in the PC and digital consumer device markets.

Storage technologies, driven by the computer industry, are cascading in price and capacity by factors that are almost unimaginable. In the last ten years there has been a one hundred-fold increase in the capacity/cost ratio of hard disk drives and currently the ratio is doubling every ten months. Incredibly, without any technological revolutions, this trend is reliably projected to continue, even to accelerate, for at least the next ten years.

In the year 2000, 10 GB of hard disk storage should retail at around $100, providing some four hours of audio-visual storage of MPEG-2 material at 5.5Mbit/s.

Assuming disk capacity/price doubling every 10 months and a more pessimistic 18 months, the following capacities at 5.5Mbit/s should be available for a retail cost of $100: -

Year	@ 18 months	@ 10 months
2000	4 hours	4 hours
2005	40 hours	240 hours
2010	400 hours	14,400 hours

Video content stored on disk for $100

Although today at a relatively high cost for the time stored in comparison with analogue tape, such disk storage will provide secure broadcast-quality storage, instant and random access, simultaneous record/play capability and will offer the opportunity for very simple control by users and agent technologies alike.

This document describes a number of applications which can be realized through the provision of home storage systems, made easy to use by appropriate use of content description, markers, links and agent technologies.

Material may be loaded onto a home storage device from broadcast content in real time, and also in non-real time, when the broadcast content is intended solely for recording. Material may also be loaded onto the home storage device from a remote storage system, in the manner of a *virtual video shop* using a video file transfer. The faster the connection, the more quickly the content will be available for consumption.

Disk storage will provide the ideal solution for time-shifting material, it will permit the viewer to start watching content before the recording has completed or to take a short break in the middle of live content without missing anything. It will also allow users to view material in a non-linear fashion, moving forwards and backwards easily with the ability to use defined index points and links where these are incorporated within content which has been so designed.

Tape storage systems may supplement disk, offering large volume random access storage. Also, there will be always be a requirement for content to be transferred from on-line storage to low-cost bulk media to be placed on a shelf for future consumption or to be passed to others.

The descriptive applications also consider the relationships between different media types, and the need, for example, for a storage device to manage a set of bookmarked WEB references obtained while consuming TV content.

Agent technologies will enable all of the above to be accomplished simply and easily; the agent making decisions about what should be recorded or deleted, managing the storage space according to the user's expressed wishes or previous behavior, based on information about the content being supplied or on offer.

The capability of IP-based and other systems to carry audio-visual material in a digital form creates the opportunity to deliver AV material to any location which has a connection of sufficient capacity and quality. This enables users to access services and content from remote locations. Services considered here encompass television, and audio-only services from high-quality music through to low bit-rate speech.

3.2 Key Concepts

3.2.1 Source content is located in time and space

A fundamental assumption in the development of these applications is the temporal and spatial nature of content, which may be acquired by the local storage and network based system. Connectivity provides the means to obtain content which is not directly available at the point of consumption, and local storage provides the means to obtain content that is available in the future. Some material may be available both remotely **and** in the future.

In the diagram below the local storage system, or the user, views the world outside on two axes - one labeled *connectivity* and the other labeled *time*. Material that is accessible using an access network - as a file to transfer into the local storage system - is located directly on the axis of *connectivity*. Similarly, material that will be delivered via broadcast channels (terrestrial, cable or satellite) to the user's premises is located directly on the axis of *time*. Material which is located on a channel which is not accessible in the user's premises, but which must be accessed via a network is separated from the user in *connectivity* and *time*.

As time progresses, material on the right-hand side of the diagram moves towards the left, eventually reaching the *time zero* line. At this point material may be transferred into the local storage system, either directly or using the access network. It may also be transferred into a remote server, and become available for future transfer as a file.

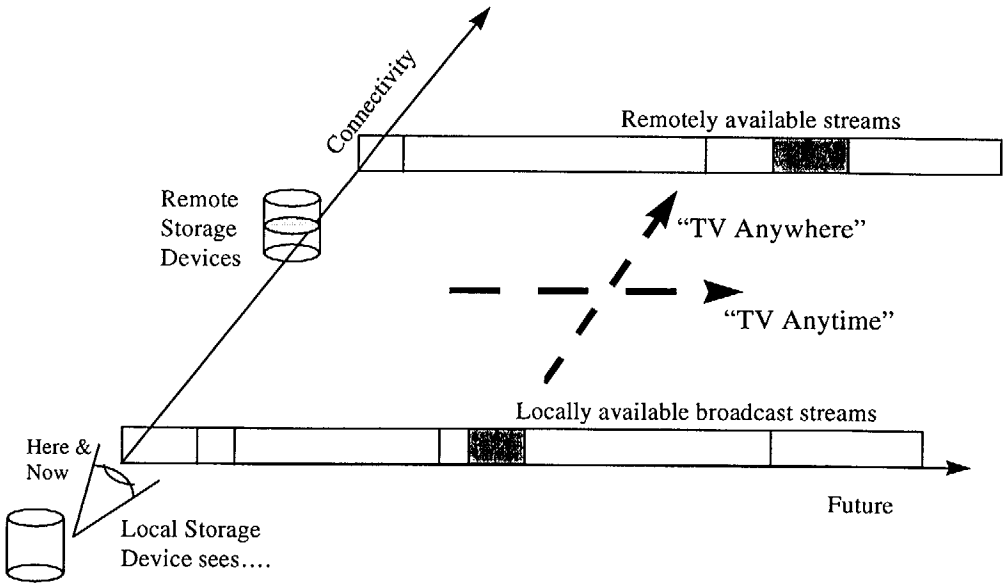


Figure 1 : Source Content is Located in Time and Space

The diagram introduces the key concepts of *TV Anytime* and *TV Anywhere*, representing the temporal and spatial separation of a particular piece of desired content from the user's local storage system.

3.2.2 Selection

The logical selection of audiovisual material consists in the identification of material that meets criteria defined by a human. Selection can involve several steps.

In practical cases, several selections can be chained from general searches producing large sets through specific searches within sets to final direct selection from the remaining list of items. Selection can be user initiated or agent initiated. Selected programs can be either a specific program or a collection like a series or a thematic set.

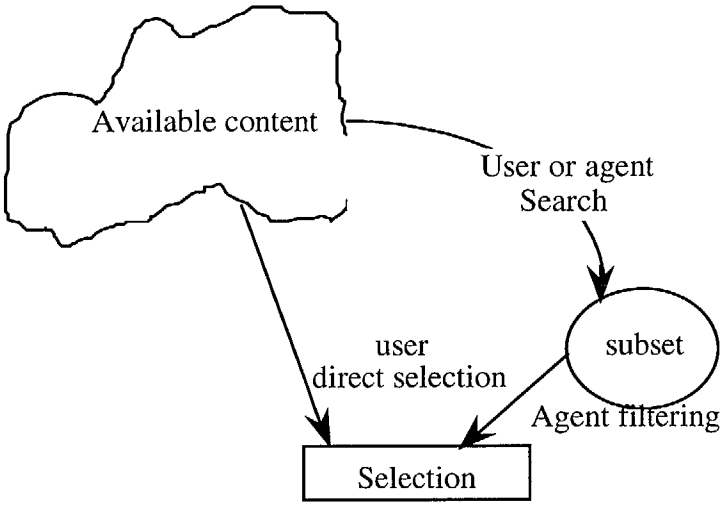


Figure 2 : Selection Process

Selection mechanisms include:

- **Search-based selection**: A search engine or an agent accepts a request and produces a list of items supposed to fit the criteria. The search may use many data sources, including EPG data or program metadata.
- **Direct selection** (e.g. activates link on a web page or chooses from a list): Direct selection may be from an EPG, from a promotion, or from other printed or electronic sources.

There may be security issues related to the search and selection mechanisms, such as:

1) Closed group items. In some cases, only the members of a special group should know of a program's existence. Others should not have their search engines list this program.
2) Parental guidance and security ratings can be viewed as a special case of 1).
3) Security and/or cost metadata could be included as search criteria.

3.2.3 Fulfillment

Having selected and identified the desired content, the process of identifying the component elements, locating them in time and space, and their subsequent acquisition, is known as 'fulfillment'. The fulfillment process may be further broken down into three successive lower level processes:

- **Decomposition:** The selection process will have resulted in a unique human-relevant or computer ID, or collection of IDs, which may expand into multiple IDs. For example, a sequence or series of programs, such as the separate episodes of a drama, or a 'collection' of programs identified as being relevant to the user's selection criteria.
- **Resolution:** Having identified the elements, it will be necessary to determine where and when these may be found. A content element might be located in the local store, in a current 'live' broadcast, be the subject of a future broadcast, or available remotely on the Internet as a file or stream, either now or at a future date. It is necessary to keep track of any changes of location and/or timing of the required items; for example, a sports event taking longer than expected.

- **Acquisition:** Finally, having verified the user's rights to acquire the content, appropriate action is taken to obtain and store the content elements on the local storage device.

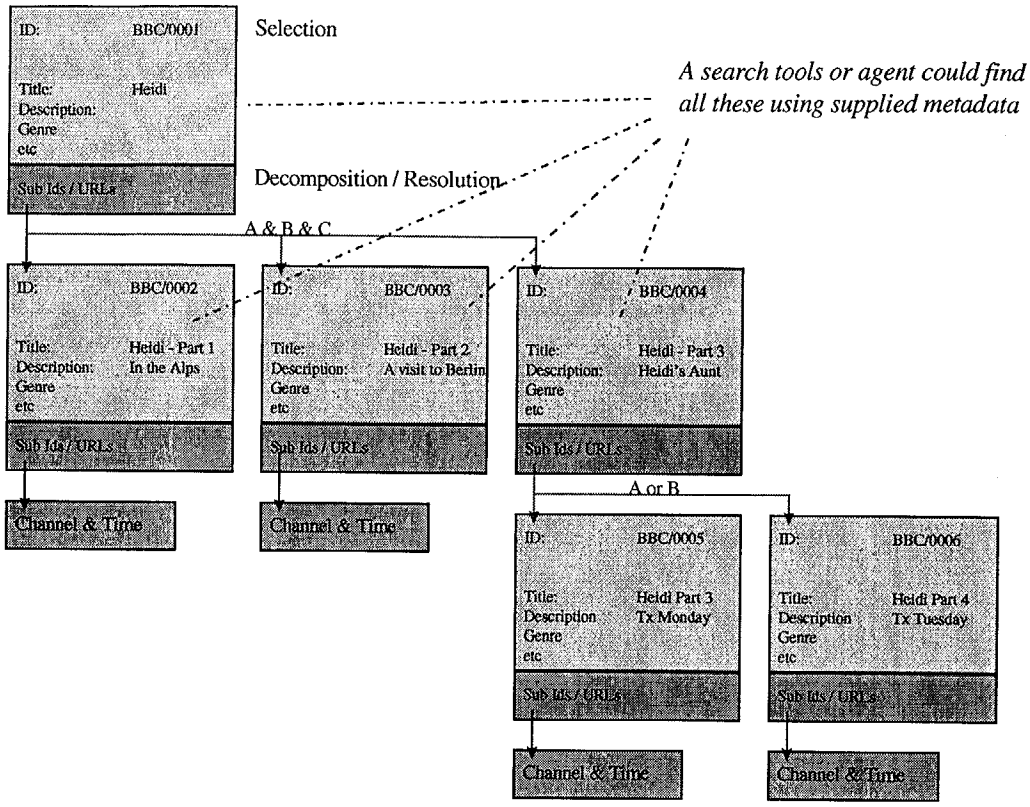


Figure 3 : Illustration of Required Processes of Selection, Decomposition and Resolution

When a specific item of interest is selected, the system will be given a unique identifier that represents a persistent reference ID to the content, which is independent of time and space of delivery.

The resolving authority for this ID will then provide a mechanism by which the ID can be resolved automatically into a physical locator who specifies the time and place of broadcast. In some cases this process may require several steps. In the example Figure 3:, the series *Heidi* identified by *BBC/0001*. This is a collective ID, which decomposes into three episodes. When *BBC/0001* has been selected for recording, all three episodes will be captured automatically.

For each program item to be captured at least one physical locator is eventually required. In the case where multiple locators exist (e.g. repeated transmissions of the same content, where one locator or another may be used) the locator to be used may be chosen according to criteria such as time, cost, quality, or conflict with other content recording. In Figure 3:, Episode 3 of the *Heidi* series, identified by *BBC/0004*, resolves into *BBC/0005* or *BBC/0006*. Either may be used by the recording system.

While unique IDs can reference complex program structures based on relational links between content items, the search and filtering tools may deliver to the capture or fulfillment process a single ID corresponding to any point in the decomposition chain.

Once given to the capture process, fulfillment will result in the capture of all content items hierarchically decomposed from that point.

If a user or agent wishes to move up the decomposition chain – say where a user is attracted to capture a series having seen a single episode – mechanisms will need to be provided to allow users to derive any collective IDs from the resolving authority. Alternatively, the broadcaster may wish to insert explicit links to the series or any other related content into the content itself, using the same mechanism described for promos and trailers.

3.2.4 Management and Use of Acquired Content

TV Anytime and *TV Anywhere* will provide the most benefit if the local consumer device contains rewriteable storage for more than a single program. If the acquired material is stored, it may be viewed at the convenience of the user. Depending on the rights conferred by the service provider, it may be viewed more than once or saved in longer term, off line storage. Most consumer devices with significant storage capacity will require one or more means for the user to selectively delete material.

When content is wholly or partially stored, then the content may contain structure to be easily exploited by the consumer device. A simple example is embedded links within the content to suggest alternate material of interest to the viewer. This could be references to material in local storage, or to material stored on publicly accessible networks. Such links would be invoked upon the explicit request of a viewer. It is essential that this process is graceful, easy, and intuitive for the user without the cognitive or interface complexity more typical of today's public networks.

A more complex example requires that content be developed with specific reference points within it and suitable linking mechanisms. The reference points, having been provided by the content producer, will surely be designed to be compatible with the content producer's commercial interests. One intent of such a structure permits the viewer to see, for example, the weather and economic news first and skip the sports. An elaborate example envisions content specifically developed to permit variety in the viewer's experience. This may include customized advertisements to be viewed on the basis of geographical location or viewer age, insert points to permit customized children's programming to the name(s) of the viewing children, or alternative content pre-transmitted for display at the appropriate time.

Management of consumer device storage space is important. The consumer device owner may wish to manage space among several family members or for different types of recorded material. Management may require policies for deleting material to make room for new content, for archiving material, for restricting viewing of selected material, etc. The general case of space management is very complex and may be difficult to implement while maintaining simple and intuitive interfaces for the user.

If the storage cost is high, a consumer device supplier may lower the consumer cost of a box in return for the right to use part of the local non-volatile storage. Example uses of such a right could include download of catalogs or web-like content with commercial benefits for the supplier. The storage rights granted to the supplier then must be ensured in competition with the needs and desires of the user.

3.3 Application Scenarios

A number of example application scenarios are developed in this document to describe the functions of a home storage based system, and indicate requirements of the underlying technologies, protocols and data elements.

The scenarios are classified into six categories.

- Broadcast Capture
- Video File Transfer
- Remote Stream Access
- Web Links
- Segment Jumping, and
- Content Customization

The first three categories include applications for locating and capturing content into local storage.

The last three categories include applications that can be performed after the content has been captured.

The presence of local (home) storage provides new opportunities for usage of audio-visual material in the home. Traditionally, this material would be viewed sequentially either live or from a recorded tape. If home storage includes devices that permit non-linear access to the content and include some degree of processing power, then many new forms of content use are possible.

Each scenario is reviewed briefly in the following sections:

3.3.1 Broadcast Capture

3.3.1.1 Scope

Broadcast capture applies to services and applications that are able to effect the automatic *recording* or storage of broadcast content for use by the user sometime in the future. The storage device may be a combination of long and short term, high and low latency media. Material that is to be captured may be transmitted in real time (that is at its presentation rate) or, when specifically intended for recording, in non-real time.

Broadcast capture may be:

- User initiated, where the user selects future content directly from an electronic program guide (EPG) or from a cross-reference contained in another piece of content (for example a promo).
- Agent initiated, where the user specifies certain characteristics in advance and the agent selects and records content that meets the characteristics.

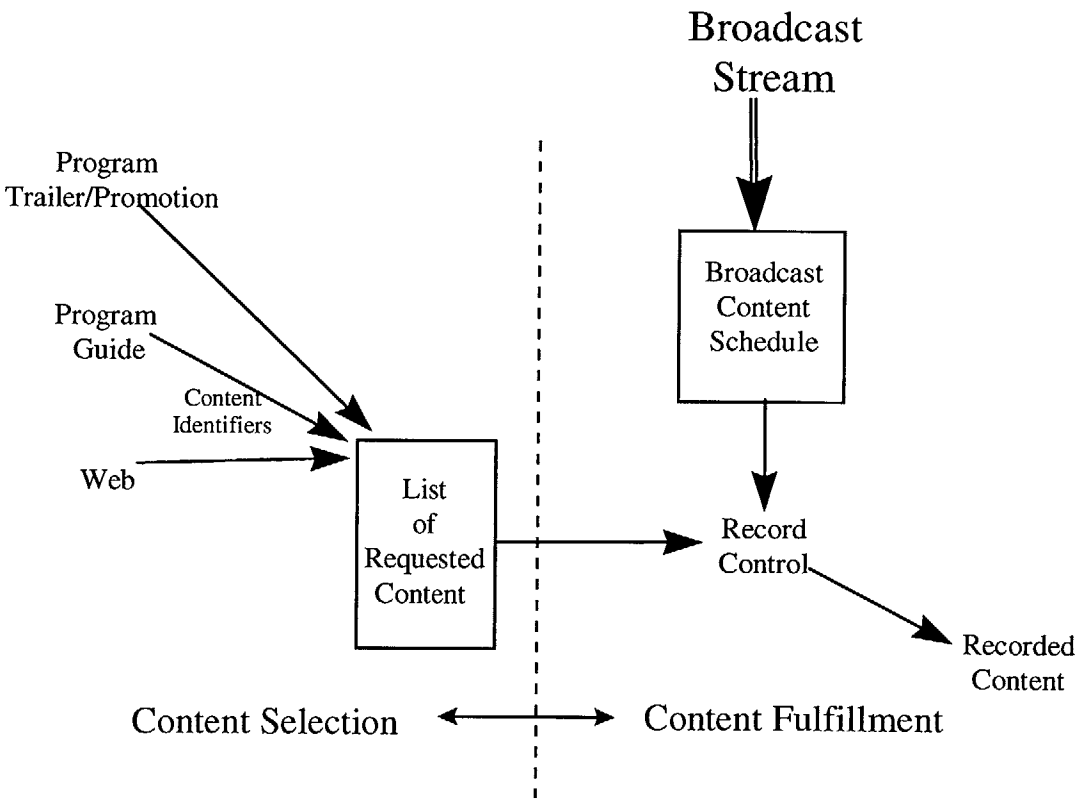


Figure 4 : Content Selection and Fulfillment Processes of Broadcast Capture

In both cases content is selected according to information (including any information relating to payment) supplied in advance. Figure 4:illustrates how the selection process leads to the acquisition of content – the fulfillment process. Selection is the process by which the user or the agent selects specific content to record, based on a link obtained from a trailer or promo, through the user reviewing descriptive information via a program guide or Web, or through the agent matching the descriptive information to a set of selection criteria. Having been identified and selected, the content is listed for acquisition by the home storage system. The fulfillment process then requires mechanisms to enable the identified content to be acquired despite changes in transmission time and channel scheduling.

The home storage system needs to manage the disk space and to identify conflicts in content acquisition (when, for example, two pieces of content from different sources are scheduled for capture at the same time). More importantly, the system should be capable of operating in parallel with normal viewing and should be able to record content whilst a viewer watches any other material.

3.3.1.2 Scenarios

3.3.1.2.1 User Initiated Capture

Four possible scenarios are identified: broadcast capture using an EPG, broadcast capture using the internet, broadcast capture using embedded references in other broadcast content, and finally the facility of organized immediate recording including the capability to support dynamic viewing.

For broadcast capture via an EPG, the user is presented with an EPG containing, say, a list of scheduled content for the next week. The EPG will need to present data which is not only derived from factual and production information but will also requires a distinct marketing edge taking into account the user's perspective of the content. With the click of the remote, the user simply selects the name of the program that interests him or her. The application will then automatically register a unique identifier for that content and ensure that the content is recorded whenever and on whatever channel the program is broadcast, regardless of any subsequent changes in program scheduling. The application will also ensure that unused space of suitable size is found for the content or will inform the user of the need to delete some other content that is no longer of interest. After capture the user will be informed in order that the user may watch it at his or her leisure.

A similar scenario may be imagined for broadcast capture using the Internet. Here, the user may be attracted to a reference on a Web page of a program to be broadcast sometime in the future. The user simply clicks on the reference link and the program's id is automatically registered using the same capture procedure of the EPG scenario.

Another scenario uses embedded references in broadcast content. A program of interest may be brought to the user's attention during a broadcast. The user may then click on these references while continuing to watch the current program in order to automatically trigger the recording of the program of interest some later. An example might be during a trailer, when a user may register his or her wish to record the program without the need to know when or on what channel the program will be broadcast. Again, recording, storage and access procedures will be identical to that of the EPG scenario.

A service provider might be enabled to load content into the home storage device as a background task in non-real time without a specific request from a user. When the content has been loaded, the user could be informed that the title is available for viewing immediately, without the need to wait. After some period of time the content could be erased and replaced. The user could be charged for viewing the title only if it is actually watched.

Finally, there is the prospect of *immediate* recording. The user may wish to record a current program *on the spur of the moment* some time into the broadcast. Today, VCR technology requires that the user first find a free tape, establish that there is sufficient room available on the tape to hold the program, press the record button and hope for the best. With a home storage system implementing *immediate* recording, the user simply presses the record button. The system then finds a free area of storage and ensures that the program is correctly recorded – including any missed beginning portion. There may also be a facility to automatically detect the end of the current program in order to terminate the recording.

A further extension of *immediate* recording might include the prospect of dynamic viewing. For example, the user is watching a broadcast television program and wants to interrupt the program viewing. Then user may perform *Pause*, *Rewind*, and *slow-motion* type operations between the *near past* and the current broadcast point. While such an application may be less reliant on the program selection type attributes required to implement broadcast capture; it is assumed that Dynamic Viewing would utilize many of the same storage management procedures. In addition, Dynamic Viewing will require a simultaneous read-write capability

3.3.1.2.2 Agent Initiated Capture

Agent initiated capture can be seen as being a logical extension of the user-initiated capture scenarios where instead of the user explicitly selecting a program for recording, the agent automatically performs the task based a user profile. As such, many of the program attributes of user-initiated capture will be directly applicable to agent initiated capture, in addition to potentially many new attributes that will be required to profile different program categories.

Agent technology may require crisper categorization than is needed for user initiation. The categorization attributes of content will need to describe at the very least commonly used genres such as *news, movie, soap* or *documentary* and, in addition, to be expandable to include new genres such as *docu-soaps*. Content can be classified according to multiple categories; e.g. the capturing of a broadcast item corresponding to a search of a

movie of one particular subject matter would not preclude the same movie be captured as a result of specifying a completely different subject matter that is still relevant to that film.

In one example agent application, on installation of the system, the user is prompted to describe his/her personal likes and dislikes in TV viewing, based on genres, categories and types that are also attributes of digital TV content. This should be done in a manner that is pleasing to the user.

The user can specify

- programs to be captured on an ongoing basis, for *Anytime Services* (e.g. latest news, specific sports, soap installments, children's programs);
- features to be captured in case they come up, such as categories of movies (e.g. horror films, specific directors);
- programs featuring specific persons;
- programs featuring specific themes (e.g., crime, politics, technology).

The user also specifies personal dislikes, programs that should never be captured, e.g. pornography. Likes and dislikes can be given priorities, prompted by the system, based on storage capacity or other constraints.

The system then filters incoming program and service information, using the attributes from Service Information, information from specific web sites etc. Based on weighted filter output programs are automatically captured on local storage. Results of this capture process are presented to the user in a manner consistent with the local EPG. This presentation may include user-specified prioritization or ordering of the available live and recorded material. The user can select captured features for viewing in the same manner as selecting real-time broadcast programs.

The system can also be made adaptive by a second filtering process, taking into account the actual viewing habits of the user, again using the attributes assigned to digital TV program categories. The actually viewed categories may be different from the specified likes and dislikes. By suitable weighting of the specified and adaptive filtering processes the personal profile is adjusted, possibly with prompting from the system.

The system provides for automatic management of storage capacity, using specified priorities, prompting the user when necessary, e.g. to take a decision about what to delete for release of storage capacity.

The user should pay only for the services used. For example, a pay-per-view movie stored on a local storage device but not viewed should not lead to any payments.

3.3.1.3 Constraints and Comments

The descriptive data of the content needs to be defined and also needs to correlate with user expectation. Content should use globally unique identifier. There may need to be a defined interface to and from the storage device.

The availability and use of a set of standardized program attributes is a necessary interoperability requirement to make automatic capture work. These attributes must be provided by broadcasters and service providers in the MPEG-2 TS. Basic program attributes can be found in Service Information, possibly some extensions are needed. A future possibility to make the system more independent of this is feature extraction. Procedures and mechanisms will be required to maintain the relevancy of content attributes. In particular, a means must be found to deal with changing and evolving categories of content; e.g. content attributes will need to be extendable to include future new content attributes.

The system presumes a digital TV consumer device, a set of filtering mechanisms, local storage and an alluring user interface.

Automatic capture assumes some of the capture services tools offered by the broadcast capture scenario, the agent taking the place of the user.

3.3.2 File Transfer from Remote Storage Devices

3.3.2.1 Scope

This application provides the user with the capability to receive and store video files in non-real-time and to display them later.

3.3.2.2 Scenarios

In a *pull* file transfer the user selects a title using a graphical user interface from a list of available titles to be viewed at a later time. The waiting time before the title may be viewed is variable, ranging from say almost instantaneous access to a very slow download operation. High-speed delivery may be offered as an extra cost option or possibly in exchange for lower picture quality. When delivery of the title is complete the viewer is notified, either on-screen or by means of an indicator light. After delivery is complete, the title may be viewed as if it were on videotape, including pause, rewind and fast forward functions. Different distribution and payment models should be supported, including:

- One time payment for unlimited use
- Period viewing
- Payment for a fixed number of viewings

Video file transfer may also be used in a *push* mode where a file is loaded onto the local disk without any action on the part of the user. Examples include video e-mail addressed to the user or movies loaded onto the disk during off-peak times. The user would be notified via an on-screen message that these had been loaded and invited to select and view them.

3.3.3 Remote Stream Access

3.3.3.1 Scope

The continuing evolution of telecommunications networks and connectivity can be expected in due course to lead to a dramatic increase in freedom for the user to gain access to material, irrespective of the physical or geographical location of either the user or the material. This will open up many more exciting possibilities for viewing and/or retrieving broadcast services and program content, unconstrained by the locations in which they are broadcast traditionally or stored.

The process of selection and fulfillment would be initiated either by direct user selection or by use of a search agent using the concepts described in Section 2. Content may be selected through a link on a web page or through a bookmarked favorite location. The user might have to select an appropriate level of service that corresponds to quality/cost requirements and available bandwidth. A connection would then be made from the user terminal equipment (maybe a PC) to the home service provider. Content may be watched in real time, or alternatively by retrieving it either in real time or in non-real time and storing locally.

3.3.3.2 Scenarios

3.3.3.2.1 Home Country from Hotel

A user on business abroad wishes to watch a home TV channel for an important event. To do so, the user selects a service, for example through a link on a WEB page or via a bookmarked favorite and a connection is made from the terminal equipment (for the traveler, usually a PC) to the home service provider. It may well be necessary to select between a number of possible services and to establish identity and/or location, in some way to gain access to the selected service. The establishment of user identity may require user authentication as well as protection of user privacy. The local terminal may also need to be authenticated and its location verified. The user will probably need to request a level of service that corresponds to quality/cost requirements and available bandwidth. The requested TV service is then streamed to the user's equipment at the requested or available quality. Using Home Storage-like concepts for caching may improve QoS. The equipment may need to resolve problems relating to different content formats (e.g. format and line standards).

3.3.3.2.2 Foreign Broadcast from Home

A user at home may wish to use a remote connection to watch TV services originating from far-distant networks. Motives may include the scheduling of special events at the user's place of birth, special events connected to the user's interests, etc. Accessing foreign broadcasts also enables people to venture out of their normal TV environment, participating in programs that are not available in their demographic local area, or to visit new regions through native TV programs. These services and applications are enabled by the connectivity offered by the global telecommunications network.

Persons may view the content in real time, which may have to be done at abnormal hours in case of a large difference in time zone, or capture content on local storage. The terminal equipment can be a digital TV receiver, a STB or a PC. Selection of programs uses the same user-initiated and/or agent initiated mechanisms as described under broadcast capture. Instead of a local EPG, broadcasters' or other web sites will help identify and locate programs.

Because the remote access could demand significant communication resources, this service may be appropriate for offering different pricing that trades viewer quality versus cost. The access, QoS and content format issues are similar to those mentioned in the Home Country from a Hotel scenario. The Quality of Service offered by the network to the user may be unpredictable and variable.

Material intended for this type of access may be designed with auxiliary languages to encourage viewers who do not speak the language of the content. It may also require novel arrangements to accommodate the interests of content rights holders, advertisers, local broadcasting regulators and others. (See Section 3.3.3 for more information.)

3.3.3.2.3 Special Narrowcast

A broadcaster may provide the service of making a specific program available on the Internet, say, after the actual broadcast has taken place. The availability may be more or less limited in time, according to the broadcaster's wishes and user demand. The storage function is provided at the service provider end.

3.3.3.2.4 Mobile Access

With the advent of global networks and wireless communication systems it is now conceivable that A/V content (broadcast and non-broadcast content) be accessible from any location regardless of physical locations over time. In other words, content can be retrieved and viewed as the A/V receiving unit moves dynamically from location to location over time. A user of a mobile A/V system may wish to view their favored broadcast program while travelling to and from work or while on vacation. This would mean the broadcast content should be accessible via independent wireless networks. If the traveler ventures beyond the reaches of one wireless network the content should be seamlessly accessible from another wireless network. An example of this would be similar to cellular telephone networks.

3.3.3.3 Constraints and Comments

Almost all *TV Anywhere* scenarios require a re-assessment of the implications of contractual agreements between content owners, broadcasters, and viewers. Many of these agreements contain constraints on who may view, where, geographically, they are permitted to view, or when the viewing is permitted. Each of these constraints may require enforcement mechanisms in a *TV Anywhere* system or the contractual requirement may require modification to permit effective deployment of the service.

- **Personal Constraints:** Many countries restrict distribution of material by a person's age or social status. As an example, controversies over pornographic material commonly discuss the issue of whether the laws of the distribution location, the viewing location, or the laws of countries through which the material transits apply. Some countries restrict material of political or polemic nature. In most analog distribution infrastructures, the physical point of termination of a cable or transmit or receive antenna is sufficient to enforce such constraints. *TV Anywhere* technology may require means to determine who is requesting viewing, who is providing it, and maybe also the geographic location of end points or communication routes to enforce applicable regulations.
- **Geographic Constraints:** In some countries, viewers pay for the right to receive content and the content owners then restrict distribution to within country and to the licensed users. If such users are now in a different geographic territory, one can ask whether their original right to receive the content supersedes or is subordinate to the geographic distribution limitations of the content owner. A variation of this consideration can apply even if the viewer has no purchased right to the material. An example is content that contains geographically distinct advertisements, possibly providing a commercial offer that is valid only in a limited territory. If such material is not permitted to be distributed outside this territory then either content must be customized for distribution or some access authorization mechanism is required.
- **Temporal Constraints:** Film distribution worldwide is conventionally timed differently at different locations. Recent controversies over DVD encoding techniques are evidence that preservation of

this business model is important to the industry. If such material were now available via *TV Anywhere*, one must ask if the home (native) location of the person applies, or the viewing location, or the location of the distributor. Whatever the answer, means for determining the relevant location would be required or some current contractual distribution obligations would require revision.

TV Anywhere by its very nature must contain some discovery process to locate the requested material. If the experience of the Internet is applicable, then the search may locate several different instances of the same program (e.g. from mirror sites). In this case, selection of the site delivering maximum QoS would be desirable, but the choice may also be affected by applicable regulations as reviewed above.

If viewers come to see *TV Anywhere* as a logical equivalent to local TV service, this then suggests that the functions described in the broadcast capture scenarios of *TV Anytime* may be expected for *TV Anywhere*. Because many properties of *TV Anytime* are designed to support material with links to content available either on the broadcast or on locally accessible internet sites, this facility would be expected to work in a *TV Anywhere* situation.

3.3.4 Web Links

3.3.4.1 Scope

The Web Links application allows broadcast television content to provide the user with links to Web pages, to save links for future use, to manage a list of such links, enabling a DAVIC terminal or connected equipment to launch a Web browser to access the related content.

Web links may refer to pages accessible via conventional Internet means, to pages co-carried in the program broadcast stream, or to pages carried in a separate broadcast stream such as a data carousel. Because Web technology permits pages to contain multiple media types, formatted in a display-independent manner, this mechanism can provide substantial customization of the viewing experience.

3.3.4.2 Scenarios

3.3.4.2.1 Immediate Link Following

The user is watching a broadcast television program and a message appears on the screen indicating that additional information is available at a Web page. This message gives the user the options to either go to the Web site immediately (if the terminal is suitably equipped) or to store the link in a bookmark type of list for viewing later. The user selects the option to store it. The link is stored in a list and the message disappears from the screen.

Another broadcast may include links to material previously stored on local storage. This would support enriched content that would encourage the viewer to explore the ancillary material while the original broadcast continues to be recorded. When the user finishes viewing the ancillary material on web pages, links could return the user to the program location originally left or to any indexed point in the program received so far. This latter facility would require the technology to support segment jumping described below.

3.3.4.2.2 Automated Link Following

The user has selected an option to automatically save all web links to the bookmarks list without asking the user. The user can watch the program undisturbed by on-screen messages. Afterwards the user can view the bookmarks list and use the saved web links. These may well be organized according to program boundaries and associated program text. If the program had been recorded and the segment jumping facility is available, the web pages can contain references back to the recorded audio/video material.

A second example might be a sports fan who chooses to always follow links to complete game statistics whenever available during sports telecasts. Such a link following could be automatic, with a return to the live broadcast whenever the user commands.

3.3.4.3 Constraints and Comments

A full capability Web Links application requires a digital broadcast television receiver that is capable of Internet connectivity and presenting Internet content. However, consumer devices that do not have any direct Internet connectivity will still be enriched if the web pages were co-carried within the broadcast or pre-broadcast and

stored in the receiver. If local storage is not used or if the web pages are not co-carried or pre-broadcast, the user can still follow the links if they can view the stored URLs.

3.3.5 Segment jumping

3.3.5.1 Scope

Segment jumping is non-linear use of linear transmitted broadcast material locally stored on an in-home storage device. To use the material in this way, the material must be supplied with embedded links or markers such as index points and hyperlinks or a table of content. Segment jumping may no only provide inter-segment maneuverability for the viewer, but can also support cross-references between video and non-video material.

3.3.5.2 Scenarios

3.3.5.2.1 Manual Segment Jumping

With content such as magazine shows, news or current affairs, content can contain links or markers that demarcate the natural components of the content. After the content is recorded on the home storage device, it is then possible to jump forward and backwards through the material using the links or markers. To facilitate this use, content may include a table of contents that the consumer device could display to facilitate use by the usual remote control

In addition to simple indices and links, content may also include *hot* links between items. For example, an educational program might display icons or text (e.g., in the form of a question) giving options to go to related topics within the content such that user's viewing perception of the content is that of an integrative experience. In this guise, the use of the broadcast content resembles the use of hyperlinked multi-media often sold today on CD-ROMs or used in electronic documents with embedded links. If the links refer to web pages, either on the actual Internet or previously stored in local storage, the broadcaster can enrich the video experience with web material as well.

3.3.5.2.2 Automated Segment Jumping

If content is marked and indexed for manual segment jumping, the facility can be useful both to the broadcaster and to the viewer. Segment jumping is one way to automate content customization by setting the consumer device to automatically follow some links. Such automated methods may include a user-specified sequence in which a nightly news story is viewed (e.g., weather first, followed by sports, national news, financial news, etc.) In another use mode, the user could command that some links with identifiable attributes are always skipped. For example, one could always skip the business news. This use type implies that the link destinations are attributed (e.g. "weather report") and that the user has a means of identifying such destinations by these attributes.

A broadcaster may also include instructions for segment jumping. For example, for ad customizations receivers in a certain geographic area should follow one set of links, and receivers in another area another set.

3.3.5.3 Constraints and Comments

Segment jumping must have an acceptable default behavior if the local consumer device either does not implement the facility or if it has not recorded the content and link following is not feasible. The broadcaster may wish to include paid links for subscribers, and free or clear links for non-subscribers. This means that access control could be required for components of content in addition to whatever is required for the overall content.

3.3.6 Content Customization

3.3.6.1 Scope

Content customization describes a means to present audio-video content with element insertion, deletion or adaptation that depends upon a local consumer device attribute and for which the inserted element may be retrieved from a home storage device. One means to achieve this effect is via automated segment jumping described in the previous section. The particular difference envisioned in this application is a more elaborate

ability to identify material by the nature of its attributes and make decisions based on those attributes and local information in the receiver.

3.3.6.2 Scenarios

There are many potential applications for customizing content at the home, rather than at the service provider. Such customization could include adaptations based on specific attributes of the viewer, or could permit the content to include material either pre-recorded or distributed previously to the home. The customization depends upon technology such as markers (such as required for segment jumping) and links (such as required for Web Links) and enhances these with a decisioning capability to choose links based on locally unique information.

3.3.6.2.1 Locality Customization

A consumer device has its postal code or another locality attribute defined. Content with embedded alternative elements is broadcast in multiple versions on one or more program streams and the local consumer device is instructed to select those elements marked for the local attribute of the consumer device. When the proper time occurs, the locally customized element substitutes for the generally available element in the broadcast stream. The capability could customize advertisements (e.g., to provide the local phone number of the nearest vendor of the product) or permit local or regional news highlight inserts (e.g., high school news coverage). It could also support crawling text for localized emergency information such as regional brush fires.

3.3.6.2.2 Person Class Customization

A consumer device has stored an attribute of the person watching. One use would be the age or gender of children that use the particular consumer device. Content with embedded alternative elements is broadcast in multiple versions on one or more program streams and the local consumer device instructed to select those elements marked for the local attribute. When the proper time occurs, the locally customized element substitutes for the generally available element in the broadcast stream. Program inserts might provide altered program ratings or ads customized to the gender, age or other preferences of person's watching. Another use might be to automatically invoke alternate foreign language audio for advertisements.

3.3.6.2.3 Person Customization

A consumer device has stored the name of the person watching. Content with embedded alternative elements is broadcast in multiple versions on one or more program streams and the local consumer device instructed to adapt marked elements for the name of the viewer. An application of this capability could be program inserts to customize children's programming to the name of the child.

3.3.6.2.4 Efficient Bandwidth Use

A broadcast channel may have bandwidth is insufficient for some content. One example is high definition ads in a standard definition program stream, or the desire to transmit several different video renditions of an ad for display based on the person's preferences as noted in a previous scenario. To achieve this, the high bandwidth content would be delivered earlier than the presentation time, using available bandwidth, and stored for playback at full integrity at the proper time. The real-time programs would contain appropriate markers to trigger the insertion of the locally recorded material. A fully developed use of such a capability might transmit extensive material in late night hours for use during the following day period. This latter type of use presumes that the broadcaster has some rights to use local storage, potentially granted in exchange for subsidization of the storage cost.

3.3.6.3 Constraints and Comments

Content must be developed to enable this behavior and the local consumer device requires a means of identifying the proper element(s) to record. To work properly in consumer devices without the capability the content must be designed to have integrity if the behavior is not implemented. In addition the local consumer device must have a means of identifying when and how to insert the locally recorded element. The local receiving device would require sufficient advance notice to accommodate the latency of A/V storage to ensure timely retrieval and insertion of content elements and sufficient structure in the source material to ensure establishment and maintenance of synchronization.

3.4 Requirements

3.4.1 Content Fulfillment and Selection

3.4.1.1 Conceptual Model for Content Identification and Acquisition

To obtain *TV Anytime* and *TV Anywhere* functionality, the user must identify A/V content of interest and the DAVIC system should then determine when and where that content may be obtained. The conceptual model of this process has a selection phase followed by a fulfillment phase. During selection, the user identifies one or more content items to be obtained. The fulfillment phase uses those identifications to obtain the content for the user. Differing implementations may hide or automate many required steps (emphasizing ease of use) or may reveal them or offer choices (emphasizing flexibility), but the basic strategy remains.

Every TV program contains parts. These parts could include leaders, dramatic segments, advertisements, etc. In the fully digital TV system, auxiliary parts (such as alternate languages) or related parts (such as Web page references) may also be present. To support *TV Anytime* and *TV Anywhere*, each part must have some *name* (i.e. unique computer processable ID) that is unique to the part. The material to which the informal name *TV program* refers must have a unique name (i.e. unique computer processable ID) for the whole and may have unique names for many of its parts. Parts themselves may also have names (either a human relevant name or a (i.e. unique computer processable ID) for part components. Names (either a human relevant name or a (i.e. unique computer processable ID) may also refer to collections of programs. Unique names (i.e. unique computer processable ID) and name relationships (e.g., part-of, collective-name-for) are a foundational property of content that permits users to identify, locate, and access Audio/Visual material.

To permit a DAVIC system to identify material precisely, names (computer Ids) of collections of content, content, or content parts must be globally unique over a many year interval. Such names will inevitably be long, complex, and of no interest to users. In contrast, the names that people use may be ambiguous (e.g., there could be multiple broadcasts of the same material) or refer to a collection of material (e.g., the human relevant name of a dramatic program series). People may also name programs by their attributes (e.g. college football). Therefore, an essential property of a DAVIC *TV Anytime* or *TV Anywhere* system is the ability to translate a human-friendly ambiguous name (e.g. the 6 o'clock news) to a precise (computer ID) which can refer to a precise {channel, time, event id, transport id, etc.}. Such an expansion process may not proceed to completion when the request is initially made (e.g., a request for all future airings of a series). Thus the conceptual model does not assume that selection completes *in toto*, followed by fulfillment, but only that selection on an item-by-item basis completes before each item can be fulfilled.

3.4.1.2 Selection

The selection process begins when a person provides a human-sensible name or a name surrogate. The DAVIC system then expands these names into a collection of unique content names (computer Ids) that can be used for retrieval. The expansion process may be automatic or may require additional interaction with the person. Some systems may only use locally available EPG information, while others may use more extensive resources. Some selection methods may use attributes (such as genre) which may require access to metadata about content in addition to the name (either human relevant or computer ID) of the content. The selection process may be as simple as clicking on an icon during a promo describing the content available.

The collection of names that results may contain unique program references, or generic references to material that can only be expanded to a final, unique identifier in the future (e.g., an instruction to record all episodes of a dramatic series, some not yet be listed in the available EPG). The requirements below note particular types of useful naming methods that must be provided by a DAVIC system. Additional methods are not excluded.

Although the content naming system must accommodate names (either human relevant names or computer Ids) for program parts (e.g. a news story within a news program), this does not require all broadcast material to so categorize or name material. A DAVIC system is therefore constrained by the naming practices of content providers and distributors.

3.4.1.3 Fulfillment

Once a unique name (computer ID) is available, the DAVIC system must determine how to acquire the designated material. This requires consultation with a *resolving authority* to translate the program or content name to (e.g.,) a channel/time for broadcast material or archive/name for a request for material. In typical use, the resolving

authority may be the EPG and fulfillment may only require waiting until the proper time and then tuning to the proper channel. In other instances, fulfillment may require consultation with an external *resolving authority(s)* to translate the unique name(s) (computer Ids) to an archive and resource name for retrieval upon demand. It is possible that a name (either human relevant or computer id) may resolve to multiple instances where that material is located (e.g., repeat broadcasts). Fulfillment may also require additional consultation with the user. For example, resolved names (either human relevant or computer ID) may determine that a free, abbreviated version is available next week, or a paid, full version is available this evening.

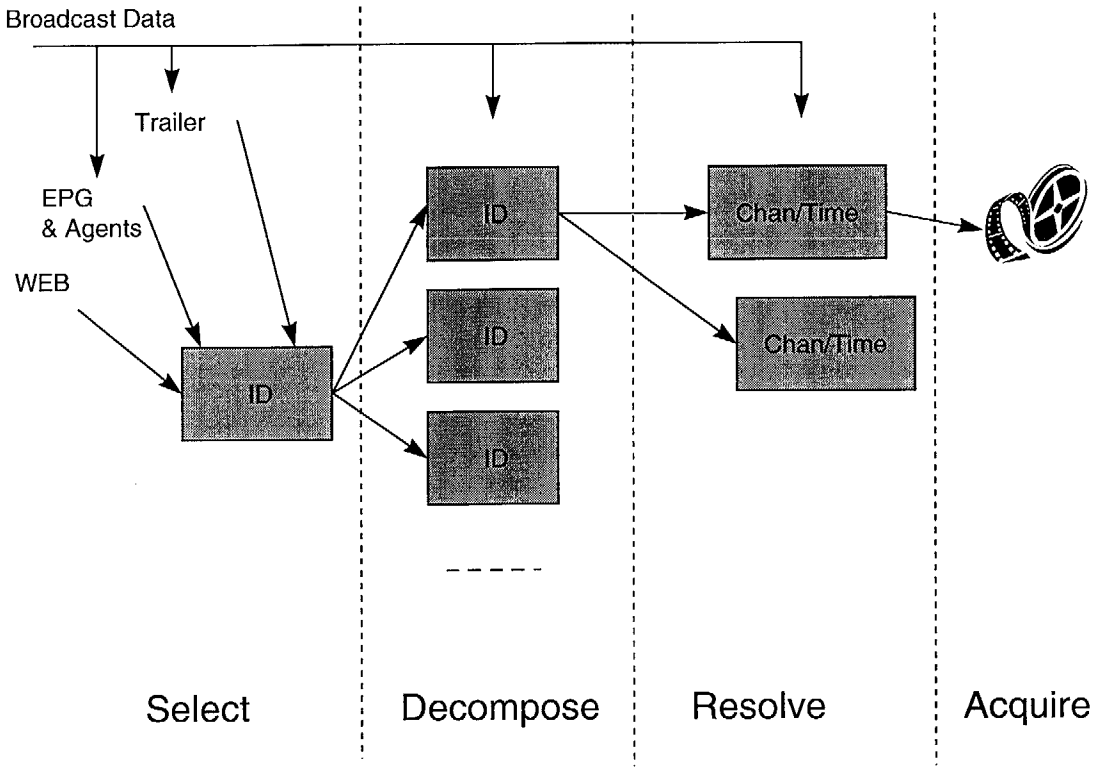


Figure 5 : Content Selection and Fulfillment

Figure 5:illustrates how the selection process results in a unique ID which may expand into multiple ID's in the case of a sequence or series of programs which again resolve into locations in space and time where the program may be acquired.

3.4.1.4 Program Identification Requirements

A DAVIC system must provide the means to map program identification into unique content identification and the means to locate the material so identified.

To enable full use of *TV Anytime* or *TV Anywhere*, a DAVIC system must provide means for users to easily:

1) identify programs without requiring use of channel and time of broadcast. Program is the material implied by the public name of the content in a customary broadcasting interval (e.g., all material presented during the evening news program, including advertisements, leaders, etc.)

2) identify the channel and time on which a program is broadcast at specificity similar to current printed guides.

3) identify user-relevant segments of a program (e.g., news story, data component, advertisement)

4) identify sequences of program material (e.g., a TV series, news broadcast series) with a single reference. The requirement is intended to permit easy reference to multiple episodes of the same dramatic series, or to successive broadcasts of closely related content (e.g., news broadcasts).

5) identify multiple instances of material (e.g., multiple broadcasts) in which the potential differences in material are not of interest to the viewer (e.g. they include different advertisements).

6) identify instances of content related to but not identical with other instances. The system should permit users to select material that has user-valued dissimilar properties (e.g., a full length vs. a movie edited for time constraints). Such content references may refer to entire programs or to program components.

7) identify program download opportunities & location if the system supports on-demand retrieval

8) identify legal or economic conditions on viewing or recording material.

If a system provides the capability to store unrequited material locally for viewing upon request, then the DAVIC system must provide a user-readable content description for the material. This requirement expects a description more informative than a brief title.

To support the capability for navigating within a program, a DAVIC system must provide means to

9) identify the current temporal position within a program

10) identify referenced content not part of the primary program (e.g., web-based material, stored material) consistent with equipment capability.

3.4.2 Content Rights and Security

The requirements for any security system deployed in a *TV Anytime* and *TV Anywhere* device will be defined according to the business demands of the content vendors and/or the broadcasters. In many cases, this will mean continued support of existing business paradigms (subscription, PPV) as well as future ones (rental, pay-per-time).

In order to facilitate a speedy development of *TV Anytime* and *TV Anywhere* systems, security mechanisms should initially focus on providing support for simpler business models, while permitting future extensions. Whatever system is chosen, it is inevitable that content protection cannot be absolutely unbreakable. Ultimately, motivated and well-equipped pirates will be able to obtain unauthorized access to content. What is important, however, is that their investments are rendered unprofitable. Piracy, with a significant initial investment will require the distribution of a large number of pirate modules (usually at a very low cost) in order to make a profit. Security tools, which prevent this type of piracy, are likely to be adequate for most cases.

"Free-to-air" content is generally broadcast unprotected. However, mechanisms that prevent unauthorized copying and distribution may still be required.

Taking all this into account, the DAVIC strategy for the protection of content in broadcast capture systems has therefore the following mutually supporting elements:

- Content authorizations, which must be clearly defined, communicated and accepted by the end user;
- An appropriate security system to prevent abuse of these authorizations.

The combination of content provider, service provider, security provider and network operator must have the ability to authorize users:

- to capture content on a local storage device, built into the STB or TV receiver or connected to a home network
- to use the captured material in a number of ways; e.g. one-time, many times, taking into account expiry dates etc.
- to transfer content to other media; e.g. with copy protection against unauthorized subsequent generations.

A DAVIC compliant end-user system must supply the means to enforce the content authorizations. In addition to being conformant with the content authorizations, the applications must be able to inform the user and the end-user system of the content authorization status. According to the business model, facilities such as free preview viewing should also be possible prior to the user being asked to pay. The application should be able to update the actual content authorization status after every use.

Security mechanisms should be put in place that protect the content across the appropriate external digital interfaces. Security levels will vary according to the opportunity for access to content via external interfaces. In the simplest case where a receiver and internal local storage together form a single integrated device without an external digital interface there may be no need for additional security measures other than [tamper proofing and] those mechanisms already in place such as pay-per-view systems. At the other extreme, systems that support the transfer of content between local storage devices and removable storage may require the most demanding of security mechanisms.

Content transfer between local storage devices – via removable storage or the home network – should be transparent to the content security system in place; i.e. the act of copying content should leave the secure properties of the content intact and operational. One could even envisage the unrestricted distribution of content so long as the content provider can still guarantee continued revenue from pay-per-view systems.

3.4.3 Device and Content Management

3.4.3.1 Storage Devices

At the core of the system there shall be a random access rewritable storage system such as a hard disk. This shall allow concurrent random access read-write capability.

Additional storage devices may also be connected directly or via an in-home network. For example, tape, optical or other removable-media devices for archiving or for passing content to others. Some removable media may be read-only, such as DVD. Other media may allow read-write (such as DVD-RAM), or be write-once only (for example, some types of DVD-ROM).

Content may be stored off-line on removable media. The system would be aware of the presence of a removable-media storage device.

The content management system should identify the various storage devices available to the system, and whether media may be written to one or more times.

3.4.3.2 Space Allocation

The system must determine the available remaining recording storage space for each attached storage device.

There shall be support for the allocation of space according to criteria required to implement *TV Anytime* and *TV Anywhere* services. Allocations may need to be made for exclusive use by individual users or for direct use by one or more content providers. For example, the system might need to manage space among several family members or for different types of recorded material. Priority criteria would need to be set for each user in a multi-user system, perhaps including storage capacity limits for individuals.

There should be the ability to assign storage to one or more content providers to receive favorable payment terms in exchange for the right for the content provider to access the pre-determined storage area. In these cases, content may be loaded into the assigned area of the storage device in non-real time and without a specific request from the user.

The system should support a means of recording where new material automatically overwrites the old; for example, a weather report that is updated at regular intervals. It should also support a means of recording where explicit permission is required for the content to be deleted.

The system should record all relevant parts of the content, for example, video, audio, metadata and any other required data streams. The system should record appropriate information about content items due to be recorded at some future possibly not yet known date and take steps to initiate the recording at the appropriate time. It would be necessary to store associated metadata and attractors for each item of recorded program content. These would be used for search and retrieval within the local storage system, and also to avoid storing duplicate content. The user should be informed when requested content items have been stored.

The methods of storage should allow synchronization to be retained between each element of the content on playback.

The system must detect the end of a current program in order to terminate recording.

3.4.3.3 Content Management

A *TV Anytime* or *TV Anywhere* system will record content for later viewing. It must provide appropriate content management services for all material recorded within it or under its management control, including:

1) Suitable records of requests for content acquisition. These records need not be maintained after the acquisition is successfully completed;

2) Suitable records of content that was recorded by this device, including suitable media and location information appropriate to the content management strategy of the device;

3) Means to identify the nature of all recorded material to facilitate human assessment of the recorded inventory. The identification information will include suitable components of content metadata and appropriate information supplied by the device and /or user;

4) Appropriate catalogs or indices to facilitate human evaluation of recorded material;

5) Means to effect automated or human-assisted decisions for content archiving or deleting. This must include means to require human concurrence before content is deleted;

6) Means to support appropriate automated or human-assisted decisions for space management, including decisions for deleting or archiving of less valuable material;

7) Means to identify and maintain intra- and inter- content links in the material. This requirement applies only to the standard linking mechanisms appropriate to DAVIC content. It does not require the device to detect or interpret proprietary linking mechanisms. Also, it only requires that the system ensure that links to material that it manages can be effected appropriately. It does not require any assurance about links to material outside the management scope of the system;

8) Means to manage material recorded on removable media. The system must have a method to identify material previously recorded by that system;

9) If the content management system manages content on removable media, it must verify the identity and nature of content on media that it recorded when that media becomes accessible by the system;

10) If the content management system manages content on removable media, it must maintain suitable information in catalogs or indices to facilitate human evaluation and location of recorded content. The system must have a means to advise the user on appropriate identification for each removable media.

3.5 Implementation Considerations

Systems that support *TV Anytime* or *TV Anywhere* will make access to content appear easy and obvious to the user. Many challenges to achieve that goal demand sophisticated systems under the care and management of the broadcaster or transport provider. Customer equipment design is as challenging, particularly because the goal is to avoid conscious management effort and to conceal any difficulties from the user insofar as possible. The challenge for the customer equipment designer this is not to merely solve problems, but to do so in a manner that is natural and graceful to the user with minimal surprises and disappointments.

Two functions are particularly difficult to do well and are pervasive across a wide range of applications:

- Ensuring Content Usability
- Graceful Allocation of Bounded Resources

The quality of providing these two functions is difficult to measure objectively but failures are likely to be very evident to the user. This section reviews some of the issues for achieving the goals, not as to provide additional requirements, but to remind system designers of anticipated tensions between desirable system properties.

3.5.1 Content Usability

Content storage facilitates the delayed viewing of content and the viewing of the content in other than linear order. In addition, storage permits content to be acquired over an extended time and then viewed by linking the separate elements together automatically. To take advantage of these new possibilities, content may be designed

to be a complex structure when it is delivered to the local storage, and this structure may be essential for the proper viewing of the content. In addition, the content may depend upon material and resources external to the content itself that may not be available during delayed playing. Some examples illustrate the range of problems to be addressed.

- Content may contain internal references in the nomenclature of the transport technology that delivers the material. These references may need interpretation to be valid when encountered during playback.
- Constraints may be imposed on the original material that must be preserved at playback time. This may include the requirement for payment when the material is viewed (rather than at record time) or requirements for copy management or copy control.
- Content that is potentially customized at playing time (e.g. per person) may require that all alternative playback possibilities be recorded so that the customization can be enacted at playback time. Such customization may also require support from security resources that could find the proper authorization mechanisms unavailable at playback time.
- Content that has external references may not find them available at playback time. This may be particularly true for Web links that are timely such as weather or news.
- Content may be divided among many elementary streams. Suppose some of these streams carry alternative content, only one of which will be displayed depending upon some customization. Although only a relatively small fraction of the material may be required at play time, all of it should be recorded so the alternative choices at play time are feasible. This may require a much higher record rate than for a normal content stream.
- Content may reference pre-recorded content either delivered previously by the broadcast stream or via alternatives such as CD-ROMs. The recording process detects these references and takes appropriate measures to save the referenced material as well as the base content material.

The examples above illustrate the following types of challenge for delayed viewing of enriched digital content:

1) The content may have complex structure that must be preserved and processed during playback.
2) The content may have complex dependencies that are difficult or impossible to discern and preserve between record time and playback time.
3) The content may have constraints upon its use at record time that must be appropriately honored at playback time

3.5.2 Resource Use

These properties mean that *TV Anytime* and *TV Anywhere* systems must manage diverse requests for resources, determine appropriate action when resource requests exceed what is available, and communicate with the user when automatic resource management needs human assistance.

For example, *TV Anytime* requires the availability of local storage. The storage resource must be managed by the system, deciding what to record and what to discard. At this level, the problem is familiar to users used to VCR tape management. The new systems, however, will remember commands from users to record or process information at future or unknown times. Additionally, digital content is expected to sometimes include active behavior to execute either in the foreground or in the background while other viewing is in progress.

3.5.2.1 Resource Availability

A *TV Anytime* or *TV Anywhere* system contains multiple resources that must be shared. Example shared resources include storage space, available tuners, CPU cycles, the decryption/encryption system, communication equipment and displays. When a request is made to the system, it should determine whether the local resources are sufficient to honor the request before accepting it. This determination requires an accurate knowledge of resource availability when the request is executed and resources required by the request.

Resource availability limitations come from many causes.

- Permanent unavailability: this can occur by the limitations of the installed equipment or by changes or failures in the equipment.

- Conditional unavailability: this can occur because of changes in constraints imposed by the user, such as storage space allocation constraints, usage constraints during some hours, or blocking of some program types.
- Dynamic unavailability: this can occur because previous uses have claimed resources since the initial request was evaluated and not released them. Storage space claims are one example, as are tuner reservations for background tasks.

Because customer equipment will, most generally, support the viewing of multiple users, multiple claims for resources are inevitable. If these claims are for recording at a future time, and unless the resource is irrevocably reserved at the time of request, all requests may be honored. If in the course of honoring each successive request, the previous requests were not viewed and deleted, storage space exhaustion will occur even though each individual request found sufficient space when the request was made.

Not only is resource availability not always determinable when the request is first honored, it is also in general not possible to anticipate the resources required by the request until it is executed. For example, the time required to fully record a sporting event and the resulting storage space is not well known in advance. One approach to reducing this particular class of problems could be to require metadata that contains information on the probable minimum and maximum storage size of the content. A more comprehensive approach requires metadata to describe all embedded or active components that demand special resources. Even if such an approach were universally adopted, inaccuracies in the resource estimates and the variations in actual resource requirements (known after recording is complete) will continue to challenge efficient management of resources.

In summary, it is most appropriate to determine the feasibility of the request at the time of request and advise the user. If that is not possible, the system should determine feasibility before the request is to be executed. The least desirable behavior is to abandon a partially completed request or to achieve the request imperfectly. Thus the key issue for resource management is not merely to avoid deadlock over resource claims, but to provide graceful and appropriate behavior, hiding as much complexity as possible from the user.

3.5.2.2 Resource Conflicts

Although the best procedure is to not begin a request unless it can be completed, systems must also cope with unexpected conflicts over resources during the execution of a request. Partial equipment failure is a classic but infrequent cause. Other causes include storage space exhaustion by a background activity, unexpected claims on tuners by background activity or by user actions, unexpected processing claims by content that contains active material, or unexpected requests for communication resources caused by the action of the user.

As another example, it may be that a service provider subsidizes the cost of receiving equipment by claiming the right to record information in local storage. This could be used to provide speculative availability of paid programming, pre-provision of advertisements, etc. If such pre-authorized download were to begin when the user was otherwise using the tuner, decoder, etc. resources for viewing a different service provider, there is a substantial conflict over resource use.

As another example, suppose a receiver has two tuners. A previous request has claimed one tuner to record a movie while a user is watching another broadcast live using the second tuner. During that live broadcast, the user chooses to follow an embedded link that requires use of the second tuner. Does the system refuse to follow the link, terminate the recording, or request advice from the viewer?

Resource conflicts like these are particularly difficult because they often must be solved in a very brief time or the integrity of processing the content is compromised. The diversity of causes makes avoiding these problems very difficult.

3.5.2.3 Communication with the User

Part of the resource management processes may be sophisticated and well hidden from the user, but eventually problems require user attention. The challenge is to make the request for aid simple and easy to understand. Experience with personal computing equipment does not suggest this is easy to achieve.

In the general case, users should have some control over the algorithms to:

- State and modify resource use permissions
- Respond to resource reservation requests

- Arbitrate resource reservation deadlocks
- Identify and manage resource release

Even for knowledgeable computer system administrators, these questions are complex. Systems for home use must either radically simplify resource management at the probable cost of frequent inquiries to the user, or provide sophisticated resource management algorithms and then deal with the user's reaction to the inevitable puzzling behavior. Unfortunately, reacting to puzzling system behavior or to requests they do not understand, users often use radical solutions such as power cycling.

The goal for *TV Anytime* and *TV Anywhere* systems is a simple, easy to use, intuitive, and reliable device to provide easy access to content from any location and at any time. Although this section notes some difficulties to be overcome, other complex systems have achieved the goal. Telephone systems and automotive electronics are examples of complex systems that work reliably and unobtrusively. DAVIC *TV Anytime* and *TV Anywhere* systems should seek no less a goal.

4 TV Anytime and TV Anywhere -System Specification-

4.1 Introduction

The scope of this section includes the definition of tools for DAVIC TV Anytime and Anywhere systems and the system specification that describes how the defined tools shall be integrated to construct DAVIC TV Anytime and Anywhere systems.

A tool is a logical concept that represents an implementation of a set of functions. It is not intended to specify an physical design of systems. Some tools may be decomposed further into multiple component tools to provide detailed descriptions on the tools.

The section starts with defining a system model at an abstract level, which defines five principal tools as subsystems. Then it is followed by descriptions for each of the five principal tools, which includes references to a part of other relevant technical documents for further detailed specifications.

The defined system model is intended to be generic in a sense that it should be able to cover different implementations in different network environments. One implementation of the specified system may use a MPEG-2 TS based digital video broadcast system, while another implementation may use a IP-based bi-directional network. Further, the model also covers even an implementation based on the hybrid combination of different networks, e.g. MPEG-2 TS based digital video broadcast combined with a program information (meta-data) delivery over the Internet. Although the specifications allow different system implementations in different network environments, it specifies some tools, particularly high-layer tools such as APIs and content coding, to be supported by all implementations to achieve the application portability (application-level interoperability) across multiple network environments. With this approach, we can implement the DAVIC TV Anytime and TV Anywhere coping practically with the today's diversity of platforms (e.g. networks) and without losing the system's capability of sharing digital TV programs in anytime and anywhere.

.

4.2 A list of normative references

This section was merged with section 1. However, the heading is still kept for the purpose of preserving the consistency of the cross references in the document.

4.3 System Model

Figure 6 shows a system model for a DAVIC TV Anytime and TV Anywhere system. The client is defined as a logical entity that represents a system used by a consumer. Application in the client is an application code as an implementation of application scenarios described in section **3.** We define five principal services that should be used and harmonized by the client application. And we define, for each of the principal services, a server as a logical entity. It should be noted that the client and the servers are just defined as logical entities. They are not intended to represent physical units. For example, some implementations may implement all the servers on one physical machine. However, some implementations may implement each server with one physical machine per each server. The combination of servers in different networks is also considered as one possible system implementation. A typical example of such an implementation would be the combination of media-object service implemented on MPEG-2 TS based digital video broadcast and meta-data service implemented on a bi-directional IP-based network. There may also be the case where one logical server is implemented by more than one physical servers. The five principal services are described below:

Meta-data service: Meta-data service delivers to the client meta-data, i.e. information about a digital TV program. Examples of such information include the title, the keywords and the author. See section 5.1.6 for a detailed description about the DAVIC meta-data. Meta-data service is typically used in the selection process as defined in section **3.** An important element of the DAVIC meta-data is **UPI** (universal program identifier) , an identifier of a digital TV program that is time/space/protocol independent. See section 5.1.5 for detailed description about UPI. Meta-data service tool is defined as a principal tool for meta-data service. The meta-data service tool in the client side receives and decodes the DAVIC meta-data. The meta-data service tool in the server

side encodes and sends the DAVIC meta-data. The specification covers different implementations in various network environments and the delivery of meta-data may be unidirectional broadcast or on-demand delivery with an interactive network channel. See section 4.4.1 for further description of meta-data service tool.

Location resolution service: Location resolution service resolves a UPI into a set of other UPIs and/or a set of locators. Locator is defined as information that are specific to an instance of delivery service of a program, which includes the time (e.g. of the delivery service in case of broadcast), the address (e.g. of the delivery server in case of network) and the method (protocol) to be used to receive the program. When a program is composed of multiple sub-programs each of which is assigned to an independent UPI, the location service delivers UPIs of the sub-programs for the program addressed by a UPI. Location resolution service is typically used in the decomposition and the resolution process as defined in section **3**. See 5.1.5 for further detailed description on the concept of UPI and UPI/URL resolution. Location resolution tool is defined as a principal tool for location resolution service. The location resolution tool in the server side encodes and sends locators and/or UPIs. The location resolution tool in the client receives and decodes locators and/or UPIs for a UPI given from the client application. The specification covers different implementations in various network environments and the delivery of locators (or UPIs) may be unidirectional broadcast or on-demand delivery with an interactive network channel. See section 4.4.2 for further description of location resolution tool.

Content Management and Protection service: Content Management and Protection (CMP) service delivers to the client, based on the contract established between the client user and the content provider, information that are necessary to use a TV program under the terms specified by the contract. This service would typically be used in the capture process or in the use process defined in section **3.** Copyright management and protection tool is defined as a principal tool for copyright service.

Local storage service: Local storage service executes storing (saving) a digital TV program in the local storage for the client. This service is typically used in the capture process and the management process defined in section **3**. There may be a system implementation where the local storage is directly connected to a client instead of being placed in the network to be shared by multiple clients. In such system implementations, the local server does not have to be implemented. However, even in such implementations, the client shall have an implementation of local storage control that supports the APIs defined in this specification. See section 4.4.3 for further description of local storage control tool. Local storage control tool is defined as a principal tool for local storage service. Local storage control tool in the client sends request messages to register a recording list entry or to manipulate the files in the local storage. Local storage control tool in the server receives request messages from the client, processes the submitted request, and sends a response message to the client.

It should also be noted that, although typical physical implementations of a local storage server include the capability of delivering the stored contents to the client, only storing and file management should be regarded as functions of a local storage server from the view point of a logical system architecture. Delivering contents to the client should be regarded as a function of media-object service. We should regard that a local storage server is typically implemented with a media object server. (However, there still may be an implementation of media object server without being accompanied by local storage server)

Media-object service: Media-object service delivers to the client media-objects (contents or presentable objects in other words). This service is typically used in the use process defined in section 3.1. Media-object service tool is defined as a principal tool for media-object service. The media-object service tool in the client receives and decodes the delivered media-object. The media-object service tool in the server sends media-objects. The delivered media-objects may be stream objects (realtime/time-sensitive content, e.g. audio and video) or file objects (non-realtime/time-insensitive content, e.g. text, still image, application code). The tool is composed of several tools to cover various kinds of media objects. In addition. the specification covers different implementations in various network environments and the delivery of media-objects may be unidirectional broadcast or on-demand delivery with an interactive network channel. See section 4.4.4 for further description of media object service tool.

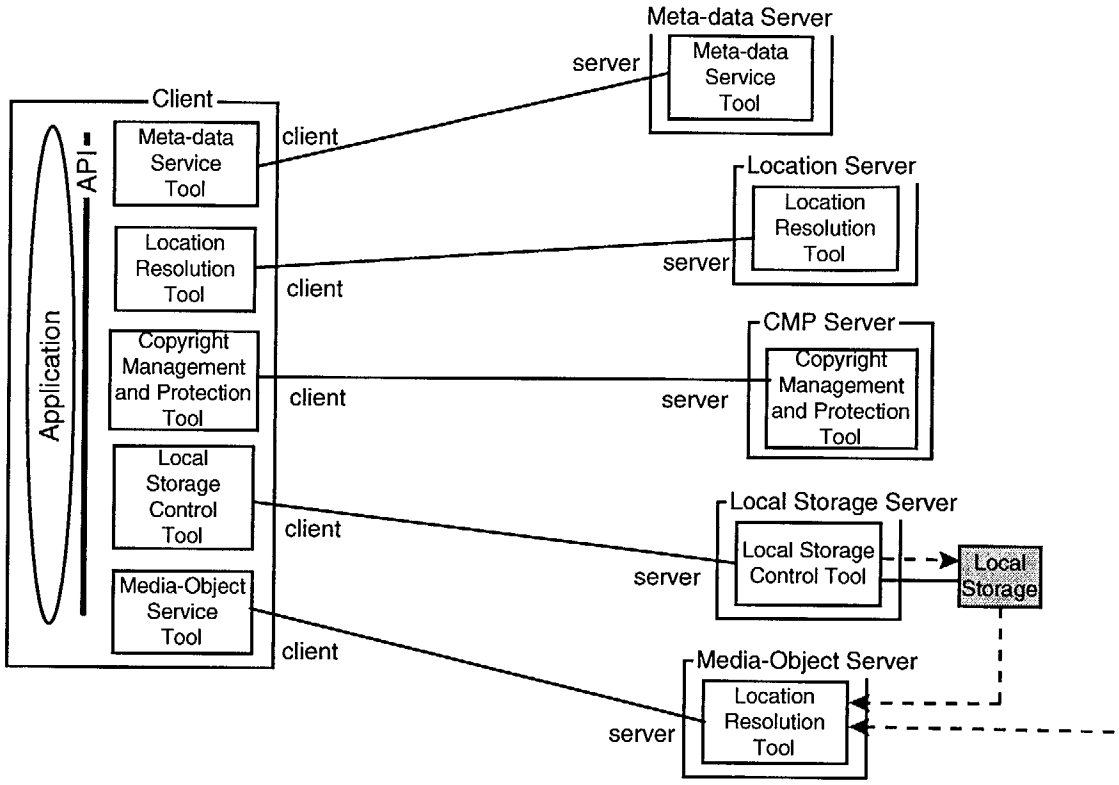


Figure 6 : System Model for DAVIC TV Anytime and TV Anywhere

Figure 7 shows main associations between the described five principal services and the processes described in section **3.** An arrow indicates services that are typically used by a process. For example, the selection process typically uses meta-data service. However, there may be another implementation of selection process that does not use meta-data service, e.g. an implementation that just displays a GUI dialogue that solicits the consumer to type in a UPI. It should also be noted there may be an implementation of an process that uses the services that are not associated with the process. For example, the de-composition process may use the meta-data service.

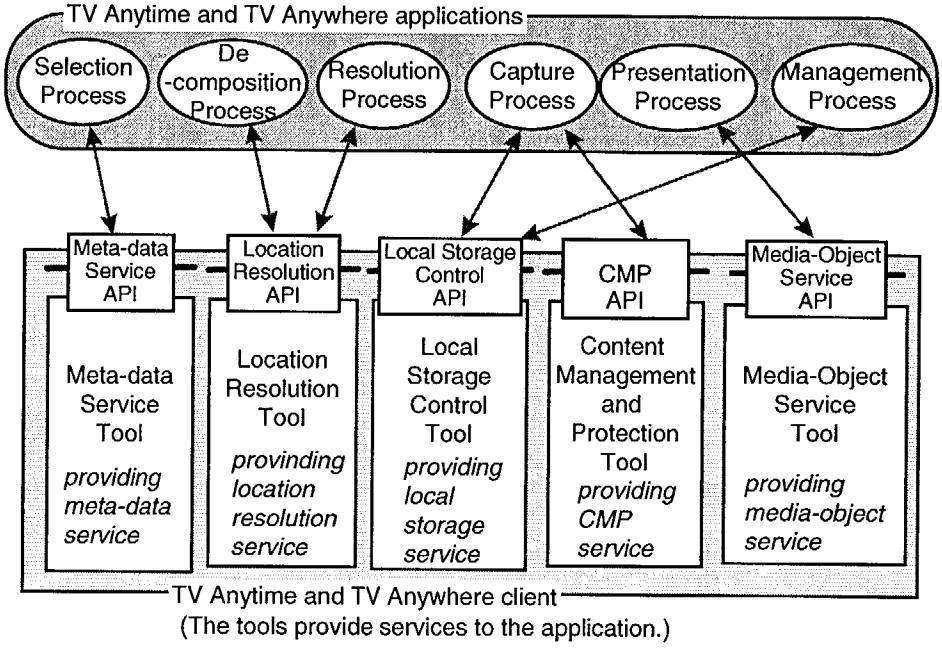


Figure 7 : Relation between application and system model

In section 2.4, we provide further specification for each of the tools defined above. Some of the tools are decomposed into multiple component tools. Table 1 gives a quick overview about the defined tools.

Table 1 Defined DAVIC tools and relevant sections

	Meta-data Service	Location Resolution Service	Content Management and Protection Service	Local Storage Service	Media-Object Service
API	Meta-data service API (section 4.4.1.1)	Location resolution service API (section 4.4.2.1)		Local storage service API (section 4.4.4.1)	Media-object service API (section 4.4.5.1)
representation tool	Meta-data schema and representation tool (section 4.4.1.2)	Locator format (section 4.4.2.2)			Content coding tool (section 4.4.5.2)
delivery tool	Meta-data delivery tool (section 4.4.1.3)	Locator delivery tool (section 4.4.2.3)			Media-object delivery tools (section 4.4.5.3)

4.4 Tool definitions and references

This section gives a further description and references to the relevant parts for each tool. Some of the tools are specified to be virtual, which means that multiple types for the tool are defined to cover implementations in various network environments.

4.4.1 Meta-data service tool

It is constructed of meta-data representation tool and meta-data delivery tool as shown in figure 8. The set of primitives for the meta-data service tool is referred to as meta-data service API.

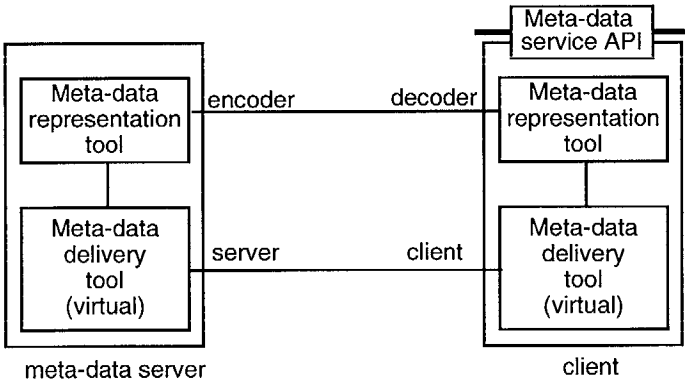


Figure 8 : Composition of meta-data service tool

4.4.1.1 Meta-data service API

DAVIC TV Anytime and TV Anywhere (I) includes no specification for meta-data service API. It may be included in the future specifications.

4.4.1.2 Meta-data representation tool

The specifications of meta-data representation tool includes the definition of the DAVIC meta-data schema and the definition of the coding format (syntax). They are specified in section 5.1.6.

DAVIC meta-data may include as its element a **program index table**, which enables indexing to particular parts of a TV program. Its information model is defined in section 5.1.4. The table shall be encoded using the descriptors defined in section 5.1.4.

In addition, DAVIC meta-data has a capability of containing UPIs and locators (URLs) that point to instances of delivery services of the addressed content. As stated in 4.4.2, the DAVIC TV Anytime and TV Anywhere (I) does not provide a complete specification for IP-based location resolution, for IP-based clients to be able to receive contents advertised via the meta-data service. Hence, it is desired that the DAVIC meta-data distributed in IP networks contain the locators that describes instances of delivery service of the addressed program.

4.4.1.3 Meta-data delivery tool

Meta-data delivery tool provides functions for the delivery of the encoded DAVIC meta-data. Meta-data delivery tool is a virtual tool, and several types for the tool are specified as shown in table 2. Implementers can select the type that suits the network environment where the system is implemented. Selecting more than one types is not precluded. DAVIC TV Anytime and TV Anywhere (I) specifies only a derived tool for MPEG-2 TS based broadcast and one for IP-based on-demand service. Future DAVIC Anytime and TV Anywhere specifications may include derived tools for other network environments.

Table 2 Derived tools for meta-data delivery tool

	uni-directional (broadcast service)	bi-directional (on-demand interactive service)
MPEG-2 TS based	MPEG-2 TS based meta-data broadcast tool	
IP based		IP based meta-data on-demand-transfer tool

a. IP-based meta-data on-demand-transfer tool

HTTP 1.1 shall be used when the encoded DAVIC meta-data is transported over a bi-directional TCP/IP connection.

DAVIC TV Anytime and Anywhere (I) does not specify a IP-based tool for the carriage of program index table.

b .MPEG-2 TS based meta-data broadcast tool

The specification of MPEG-2 TS based meta-data broadcast tool includes the definition of **meta-data package section**, which is a data section based on the MPEG-2 private section syntax designed for the carriage of the encoded DAVIC meta-data. The meta-data package section is specified in section 5.1.6.5.

Meta-data package section shall be used for the carriage of the encoded meta-data syntax over a MPEG-2 TS broadcast stream. The encoded meta-data shall be included as the char bytes of the meta-data descriptor specified in section 5.1.6.5.

The specification also includes the definition of local event information section, event relation section, PTS offset section, which are sections based on the MPEG-2 private section syntax designed for the carriage of program index table. These sections are specified in section 5.1.4. These sections shall be used for the carriage of program index table.

4.4.2 Location resolution tool

Location resolution tool is constructed of two component tools, i.e. locator format and locator delivery tool. Its set of primitives is referred to as location resolution API.

DAVIC TV Anytime and TV Anywhere (I) defines locator formats that cover both MPEG-2 TS broadcast and IP-based network delivery.

The specification also includes the locator delivery tool for MPEG-2 TS broadcast. However, the specification is still missing IP-based locator delivery tools. Future DAVIC TV Anytime and Anywhere specifications may provide complete specifications for IP-based location resolution.

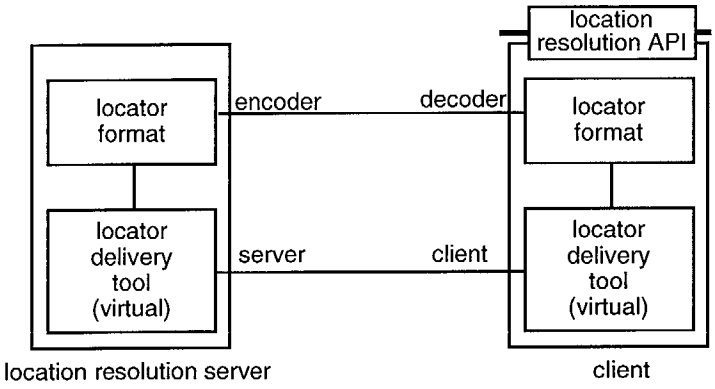


Figure 9 : Composition of location resolution tools

4.4.2.1 Location resolution API

The location resolution API shall provide the JAVA APIs for UPI resolution specified in section **5.2.4.1**.

4.4.2.2 Locator format

The locator is defined in the form of URL and is diversified depending on the types of media delivery tool. (See section 4.4.5 for the types of media delivery tool.)

In case a locator points to a MPEG-2 TS based broadcast stream using the MPEG-2 TS based stream broadcast tool defined in section 4.4.5.3, the locator shall be encoded in the form of the DVB URL specified in DAVIC 1.4.1 Part 9 Subclause 9.5.2. A capability of interpreting the defined DVB URL shall be supported by the clients supporting the MPEG-2 TS based stream broadcast tool. Such capability shall support the extensions defined in section 5.1.5.2.

In case a locator points to a MPEG-2 TS based on-demand media object service using the MPEG-2 TS based on-demand streaming tool defined in section 4.4.5.3, it shall be encoded in the form of org.davic.dsm-cc URL (DAVIC DSM-CC URL) defined in **DAVIC 1.4.1 Part 9 Subclause 9.7.1.2.** A capability of interpreting DAVIC DSM-CC URL shall be supported by the clients supporting the MPEG-2 TS based on-demand streaming.

In case the addressed delivery service is a IP-based on-demand stream service, either HTTP URL or RTSP URL shall be used. In case HTTP URL is used, the locator shall point to the SDP descriptor for the addressed service. A capability of interpreting HTTP URL, RTSP URL and SDP descriptor shall be supported by the clients supporting the IP based stream multicast tool defined in section 4.4.5.3.

In case the addressed delivery service is a IP based steam multicast service using the IP based stream multicast tool defined in section 4.4.5.3, the locator shall point to the SDP descriptor for the addressed service and the locator shall be encoded in the form of HTTP URL. A capability of interpreting the HTTP URL and the SDP descriptor shall be supported by the clients supporting the IP based stream multicast tool.

In case a locator points to a IP-based on-demand file service using the IP-based on-demand file-transfer tool defined in section 4.4.5.3, it shall be encoded in the form of HTTP URL. A capability of interpreting the HTTP URL shall be supported by the clients supporting the IP on-demand file-transfer tool.

As the DAVIC TV Anytime and TV Anywhere specification (I) does not cover file delivery over uni-directional IP networks, no locator format is defined for such file services.

The below table summarizes the locator formats adopted for DAVIC TV Anytime and TV Anywhere (I).

Table 3: a list of the locator formats adopted for DAVIC TV Anytime and TV Anywhere (I)

Media-object Delivery Service	locator		
	format	points-to	typical attributes (not exhaustive list)
MPEG-2 TS based stream broadcast	DVB URL	broadcast program (target media-object)	original_network_ Id + optional attributes listed below: transport_stream_id, service_id, component_tag, event_id, start_time and duration
MPEG-2 TS based file broadcast			
MPEG-2 TS based on-demand streaming	DAVIC DSM-CC URL	DSM-CC U-U object (target media-object)	DSM-CC Server Id, directory path and object name
MPEG-2 TS based on-demand file-transfer			
IP-based stream multicast	HTTP URL	SDP descriptor	
	SDP descriptor	IP multicast stream	IP multicast address, port numbers, start time and duration
IP-based unidirectional file-transfer	Not supported for TV Anytime and TV Anywhere (I)		
IP-based on-demand streaming	HTTP URL	SDP descriptor	
	RTSP URL	target media-object	Server hostname, directory path, file name
IP-based on-demand file-transfer	HTTP URL	target media-object	Server hostname, directory path, file name

[Note : This table is provided as a summary. Should any contradiction were found with any relevant part of the spec, this table will be modified in accordance with the relevant part.]

4.4.2.3 Locator delivery tool

Locator delivery tool is a virtual tool, and several types for the tool are specified as shown in table 4. Implementers can select the type that suits the network environment where the system is implemented. Selecting more than one types is not precluded. DAVIC TV Anytime and TV Anywhere (I) specifies only one type of implementation, which is for MPEG-2 TS based broadcast. Future DAVIC Anytime and TV Anywhere specifications may include other types of implementation for other network environments.

Table 4 Derived types of locator delivery tool

	uni-directional (broadcast service)	bi-directional (on-demand interactive service)
MPEG-2 TS based	MPEG-2 TS based locator broadcast tool	
IP based		

a .MPEG-2 TS based locator broadcast tool

The meta-data package section, defined in section 5.1.6.3, shall be used as the format for the carriage of the encoded locators over MPEG-2 TS broadcast streams. The encoded locator shall be included as the char bytes of the locator descriptor specified in section 5.1.6.3. It is not a requirement that the locator descriptors and the meta-data descriptor for a program be contained in the same meta-data package section. In other words, the locator descriptors may be broadcast separately from the meta-data descriptor for the resolved program.

4.4.3 Content management and protection tool

4.4.3.1 Content management and protection API

DAVIC TV Anytime and TV Anywhere (I) includes no specification for content management and tool. It may be included in the future specifications.

4.4.3.2 Content management and protection tool

DAVIC TV Anytime and TV Anywhere (I) includes no specification for content management and protection tool

4.4.4 Local storage control tool

Local storage control tool manipulates the recording list of the storage server and manages files stored in the local storage server. Its set of primitives is referred to as local storage control API.

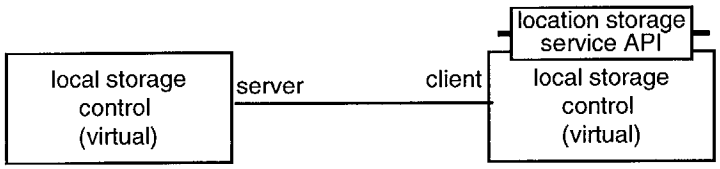


Figure 10 : local storage control tool

4.4.4.1 Local storage control API

The local storage control API shall support the JAVA APIs specified in section 5.2.6 for the manipulation of a recording list. DAVIC TV Anytime and TV Anywhere (I) does not include specifications on JAVA APIs for the manipulation of files stored in the local storage.

4.4.4.2 Local storage control tool

An implementation of local storage control tool shall support the local storage control API. As one implementation, DAVIC Intranet platform specification provides a limited local storage control capability for the live recording of streams.

4.4.5 Media-Object Service tool

Media-Object service tool is constructed of two components, i.e. content representation tool and media object delivery tool. Its set of primitives is referred to as media object service API.

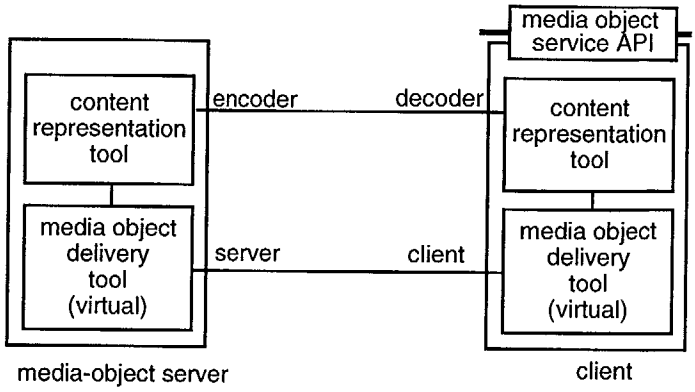


Figure 11 : Composition of media-object service tool

4.4.5.1 Media Object service API

Media-object service API shall support the JAVA APIs for media play back defined in section 5.2.5 .

4.4.5.2 Content representation tool

Content representation tool provides functions of encoding and decoding media objects. Content representation tool shall support the content representation formats described in sections 5.1.1, 5.1.2 **and** 5.1.3.

4.4.5.3 Media object delivery tool

Two types of media object delivery tools have been identified: a streaming tool and a file transfer tool. A streaming tool delivers a media object at a bit-rate that approximately matches the rate of consumption at the presentation device. It is designed such that, if the application specifies the time of presentation, the receiver can present the object at the specified time. A file transfer tool delivers a media object without any time constraint.

Each of the two tools is a virtual tool, and, for each of the tools, several types are specified. Implementers can select the type that suits the network environment where the system is implemented. Selecting more than one types is not precluded.

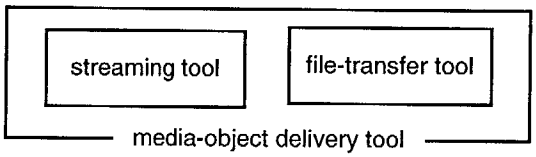


Figure 12 : Decomposition of media-object delivery tool

Streaming tool

Streaming tool is diversified as shown in the below table.

Table 5 Derived types of streaming tool

	uni-directional (broadcast service)	bi-directional (on-demand interactive service)
MPEG-2 TS based	MPEG-2 TS based stream broadcast tool	MPEG-2 TS based on-demand streaming tool
IP based	IP based stream multicast tool	IP based on-demand streaming tool

a. IP-based streaming tools

IP based streaming tool, may it be for broadcast or for on-demand interactive service, shall support RTP as defined in IETF RFC-1889.

In case a single program MPEG-2 TS is streamed, the RTP timestamp field shall be encoded as defined in IETF RFC-2250. To clarify the interpretation about the RFC, the timestamp shall represent the approximate time at which the first byte of the RTP payload is passed to the RTP layer. The amount to which the jitter in the received stream is compensated is at the discretion of the client. It is assumed that the scheme would be implemented in the network where the maximum jitter magnitude can be reasonably approximated. This assumption can be considered as reasonable in some networks, e.g. DAVIC Intranets.

For the carriage of MPEG-4 audio/visual objects over IP networks, DAVIC intends to adopt the method being specified by the MPEG-4 version 2.

IP-based on-demand streaming tool shall support the RTSP defined in IETF RFC-2326 for the control of the stream.

IP-based streaming tool in the client, may it be for broadcast or for on-demand interactive service, shall have a capability of interpreting descriptors encoded by the SDP specified in IETF RFC-2327. When the server passes to the client information about the provided streaming service, SDP shall be used as the form of information. In the case of IP-based multicast tool, the encoded descriptor shall be transported using HTTP 1.1. In the case of IP-based on-demand streaming tool, the encoded descriptor shall be transported using either HTTP 1.1 or RTSP.

IP-based on-demand streaming tool may support the RQRP scheme specified in **Jitter Concealement Tool.** The SDP descriptor for a stream service with the RQRP shall be encoded as specified in **DAVIC Intranet (I)** section 14.1. The default Audio/Visual profile defined in IETF RFC-1890 still applies.

IP-based streaming tool may optionally have a capability of reserving bandwidth for the streamed IP packets. If the tool supports this capability, it shall support the RSVP specified in IETF RFC-2205. The determination of sender TSpec is considered as an implementation matter. The tool that supports this capability should somehow obtain the information about the end-to-end (the client, the server and the network path) capability of RSVP-based QoS service. (This assumption is regarded as achievable when the system is implemented in DAVIC Intranet (I) , which assumes the clients and servers reside within a single administrative domain and hence whether the QoS service is provided or not would be a static nature of the network environment.) The tool should assume by default that the RSVP-based QoS service is not supported.

IP-based on-demand streaming tool should dynamically behave as described in **DAVIC Intranet (I) Section 16.2.** IP-based stream multicast tool should dynamically behave as described in **DAVIC Intranet (I) Section 16.3**.

b. MPEG-2 TS based streaming tools

MPEG-2 TS based streaming tool , may it be for broadcast or for on-demand interactive service, shall support the DAVIC 1.4.1 S1 flow specified in **DAVIC 1.4.1 Part 7 Clause 6**.

In addition to the above, DAVIC intends to adopt a specification for the carriage of MPEG-4 audio/video objects over MPEG-2 TS, which is being specified in Amendment 7 to ISO 13818-1 if the amendment is approved by ISO.

MPEG-2 TS based on-demand streaming tool shall support the DAVIC S2, S3 and S4 flow specification as defined in **DAVIC 1.4.1 Part 7 Clauses 7, 8 and 9**.

File-transfer tool

File-transfer tool is diversified as shown in the below table.

Table 6 Derived types of file transfer tool

	uni-directional (broadcast service)	bi-directional (on-demand interactive service)
MPEG-2 TS based	MPEG-2 TS based file broadcast tool	MPEG-2 TS based on-demand file-transfer tool
IP based		IP based on-demand file-transfer tool

c. IP based on-demand file-transfer tool

IP based on-demand file-transfer tool shall support HTTP 1.1 as specified in IETF RFC-2068.

d. MPEG-2 TS based file transfer tool

MPEG-2 TS based file transfer tool, may it be for broadcast or for on-demand interactive service, shall support the DAVIC 1.4.1 S1 flow specified in **DAVIC 1.4.1 Part 7 Clause 6.**

It is desirable that MPEG-2 TS based file broadcast tool support the DSM-CC U-U object carrousel as described in **DAVIC 1.4.1 Part 7 Clause 7.3.11.**

MPEG-2 TS based on-demand file-transfer tool shall support the DAVIC 1.4.1 DSM-CC Download as specified in **DAVIC 1.4.1 Part 7 Clause 7.3.3.**

4.5 Walk-through examples for typical TV Anytime and TV Anywhere applications

This section shows typical an example of walk-through scenarios. Typically, DAVIC TV Anytime/Anywhere applications follows the scenario shown in figure 13. For each phase, we identify the pre-assumptions for entering the phase. Some applications may not exactly follow this sequence and actual scenarios should be programmable by applications. However, whatever the sequence may be, the pre-assumptions should always be met before entering a phase.

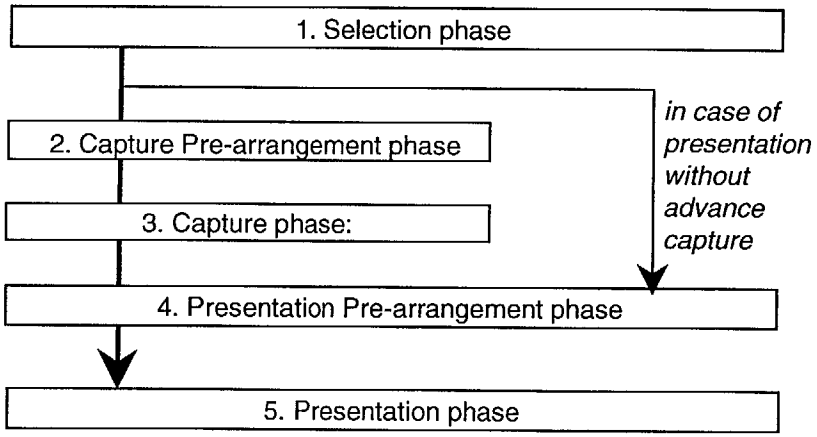


Figure 13 : Walk-through examples for typical TV Anytime and TV Anywhere applications

4.5.1 Phase 1: Selection phase

As described in section 1, in the selection phase, the client application may iterate the following processes until the program is selected.

Selection with search (see Section 1 for definition):

In the agent-initiated capture scenario, this phase is initiated automatically by the client application. In the user-initiated capture scenario, this phase is initiated when the user enters the command to receive meta-data. The

pre-assumption for entering this phase is that the client should somehow know from where and how it can receive the meta-data. In this phase, the client receives meta-data for TV programs of the user's interests. Meta-data should include at least UPI for the addressed program. Meta-data may also include the locators for the addressed program, which is desired when the system is implemented using only IP networks.

User direct selection (see Section 1 for definition):

The user of the client directly selects the desired TV program. The client application should identify the program with its UPI typically. However, the identifier may be a URL, which is likely to be the case when the system is implemented using only IP networks.

The client application should somehow know the next action to be taken about the selected program. If the next action involves capturing the selected program, the client application would go to the phase 2. If the next action is to start presenting the program without capturing it, then the client application would go directly to the phase 4.

4.5.2 Phase 2: Capture Pre-arrangement phase

This phase is initiated when a program to be captured has been selected.

The pre-assumption for entering this phase is that the TV program to be captured has already been determined and the client application recognizes it with its UPI or URL.

In case the program is recognized with its UPI, the client application invokes the **resolution** process, i.e. resolving the UPI into a set of URLs. This process may sometimes involve the **decomposition** process, i.e decomposing the UPI into a set of UPIs.

Then, the client application requests the local storage service to capture the content. The program to be captured should be indicated by means of either UPI or URL. Upon the request, the local server, or the client having an dedicated local storage device, would compute the required storage capacity directly from the meta-data, the URL or from the SDP descriptor obtained pointed to by the URL, check if the required capacity is available. If so, it would reserve the required capacity register a new recording list entry for the scheduled capture and respond to the client with a positive response. The system's behavior in the case of the shortage of storage capacity is regarded as an implementation matter.

4.5.3 Phase 3: Capture phase

This phase is initiated based on the capturing schedule registered in the phase 2. In this phase, the local storage server, or the client having an dedicated local storage device, receives the program and stores it into the local storage device.

4.5.4 Phase 4: Presentation Pre-arrangement phase

This phase is initiated when a program to be presented has been selected. The client application should somehow know if the selected program has been captured in the local storage in advance. If so, the client would reserve the necessary resources for playing out the program from the storage and go to the phase 5. Otherwise, the client would invoke the resolution process, which may involve the decomposition process as well, reserve the necessary resources for receiving the program and go to the phase 5.

4.5.5 Phase 5: Presentation phase

This phase is initiated once after the phase 4 is completed. The client presents the program. This phase would last until the program terminates or the user invokes another event.

The below figure gives an overall view of the walk-through example which is described above.

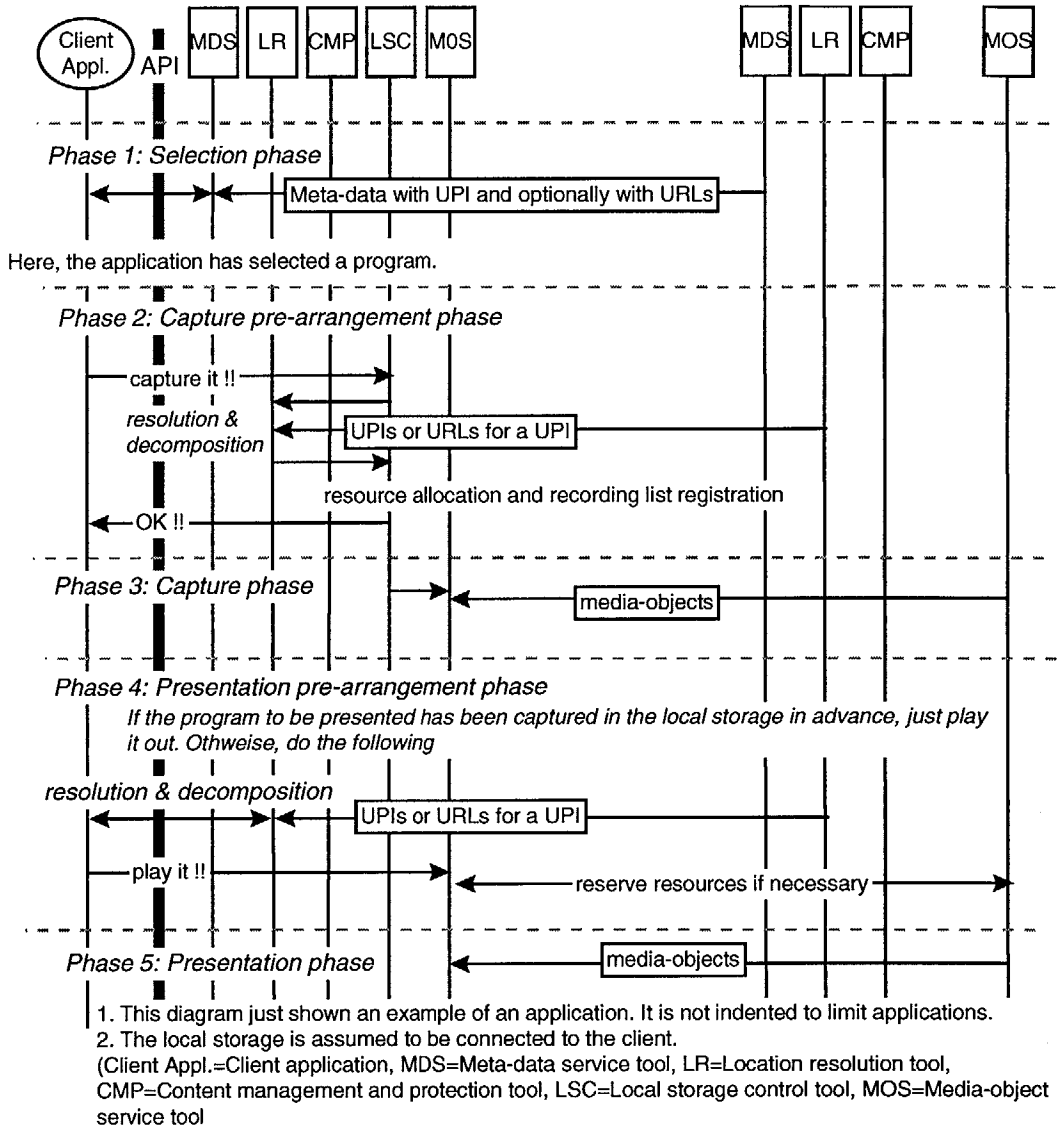


Figure 14 : Walk-through example of a typical application

5 TV Anytime and TV Anywhere -Tools-

5.1 Information Representation

5.1.1 Audio and Speech for TV Anywhere

DAVIC adopts the "scalable audio/speech profile" of MPEG-4 for broadcast over IP applications, up to level 3 as defined by MPEG-4, i.e. in mono or stereo with a maximum sampling frequency of 48 kHz. For carriage of MPEG-4 content over IP, DAVIC intends to adopt the method that is being specified for version 2 of the MPEG-4 specification.

5.1.2 Video for TV Anywhere

DAVIC adopts the "simple profile" of MPEG-4 video. DAVIC looks forward to the definition of an MPEG-4 video tool for scalable video services at a fixed temporal and spatial resolution suitable for multicast applications. DAVIC will allow future versions of DAVIC specifications to support object oriented, arbitrary shaped and VRML related (BIFS) functionality. For carriage of MPEG-4 content over IP, DAVIC intends to adopt the method that is being specified for version 2 of the MPEG-4 specification.

5.1.3 MPEG-2 broadcast programmes for TV Anytime

DAVIC specifies delivery of MPEG-2 programmes over IP using the Single Program Transport Stream format as defined in ISO 13818-1. For real-time delivery of Single Program Transport Streams DAVIC defines the use of RTP format as specified in RFC; note that RFC 2250 defines RTP timestamps indicating the time of transmission over IP of the first byte of the IP packet. The amount to which jitter of the IP network is compensated is at the discretion of the STU. However, when delivered over a DAVIC intranet, the network jitter can be assumed to be constrained to a maximum value.

For download of an MPEG-2 Single Program Transport Stream the HTTP 1.1 format is used. Future DAVIC specifications may define the download of MPEG-2 programmes in other formats such as MPEG-2 PS

There is no need for DAVIC to extend the RDM for delivery of MPEG-2 programmes over IP. Jitter removal is done prior to the RDM, while delivery to and from storage devices is beyond the scope of the RDM. In either case playback of MPEG-2 programmes is based on delivery of a valid MPEG-2 TS at the input of the RDM. In case MPEG-4 content is delivered over an MPEG-2 TS, MPEG-4 version 2 and Amendment 7 to ISO 13818-1 provide STD parameters for use in the RDM.

For download of an MPEG-2 Single Program Transport Stream over a MPEG-2 TS, DSM-CC (ISO 13818-7) is used. Future DAVIC specifications may define the download of MPEG-2 programmes in other formats such as MPEG-2 PS.

5.1.4 Program Index System

Information necessary for the programmed index system can be broadcast independently of the programs themselves. This section shows some applications realized by the Program Index System.

5.1.4.1 Program and Program Group Index

(1) Program guide

The Program Index information is displayed as a program guide screen. Program Index data is treated as a common EPG data. The information on this screen is both current and scheduled programs. If the receiver is capable of storing programs, the associated program index information can also be displayed.

(2) Program search

The Program Index can be used to specify conditions for searching a program. Current and scheduled programs can be searched. If the receiver is capable of storing programs, these can also be searched.

(3) Program reservation

A program can be reserved for viewing or recording after using (1) and/or (2).

(4) Program group guide, search and reservation

Further information of a program can be obtained, e.g. whether a programs belongs to a program group such as a series or serial. Subsequently (1) (2) or (3) can also be applied to the program group. Or you can select only some of the rerun programs (those that do not conflict with the viewing of other programs) to reserve for viewing or recording.

(5) Structured program guide

Structured Program Index information, such as a structured category of programs, can be broadcast thus enabling a structured program guide and to select a program.

5.1.4.2 Program Segment Index

The "Program Segment Index" provides the detailed description of parts of a program.

(1) Detailed program guide and search

This is similar to the Program Index described above but provides Program Segment Index to the particular program.

(2) Program segment guide and search

The Program Segment Index can be shown as the guide screen. Using the program guide index information, a program segment can be searched that matches the conditions specified.

(3) Program segment reservation

A program segment can be selectively reserved for viewing or recording. For example, only the weather segment of a news program can be reserved.

(4) Program segment selective viewing and search for beginning of the segment

In a recorded program, a segment can be selected using a guide or search function to view that particular segment of the program or to start viewing the program from the point where the segment starts. For example, a program can be viewed from a particular scene.

(5) Highlight and digest viewing

Highlights or digests can be assembled from the program segments. Normally, the segments to be used will be provided by the broadcaster as part of the index data, but it is also possible that the receiver automatically generates the segments from the index data.

For example, an audience may have to watch a live baseball broadcast from halfway through. If the index information has been sent for digest viewing, the audience can view a digest of the game first and then can watch the live broadcast. (The progress of the game must continuously be recorded.)

(6) Program restructuring and reusing

Using Program Segment Index information, a program can be restructured or reused as a source for a new program. In this process, the copyright information associated with the program must be respected.

(7) Displaying index information associated with a scene

One of the further uses of the Program Segment Index is as follows. A scene can be linked to index information. When the scene is broadcast, the index information for the scene can be displayed as a description of that scene. This can also be done while replaying a program.

5.1.4.3 Table structure used in the index system

5.1.4.3.1 Event Information Table (EIT)

Figure 15 shows the simplified structure of EIT, which is defined in DVB-SI and ARIB-SI. In the Program Index System, reference descriptor(s) may be inserted in the descriptor field in order to refer to ERT.

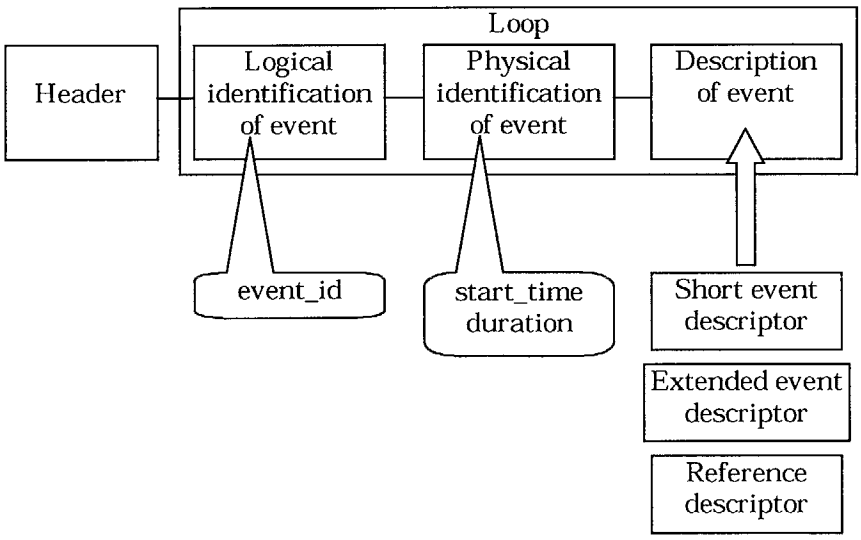


Figure 15 : Structure of EIT

5.1.4.3.2 Local event Information Table(LIT)

Figure 16 shows the simplified structure of LIT. This structure is based on that of the EIT, however, a descriptor provides the physical identification of local event. By this mechanism, the LIT is ready to support data broadcast program. Of course, EIT and LIT have different types of id fields.

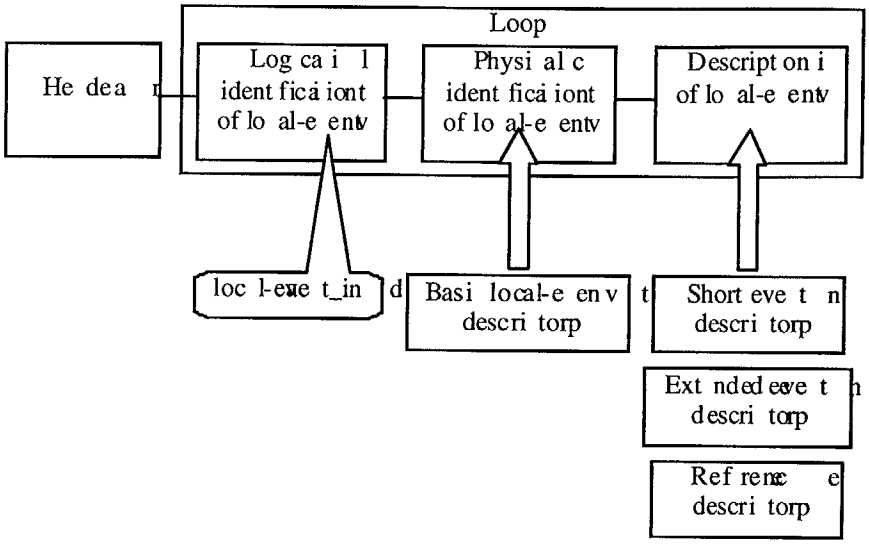


Figure 16 : Structure of LIT

5.1.4.3.3 Event Relation Table (ERT)

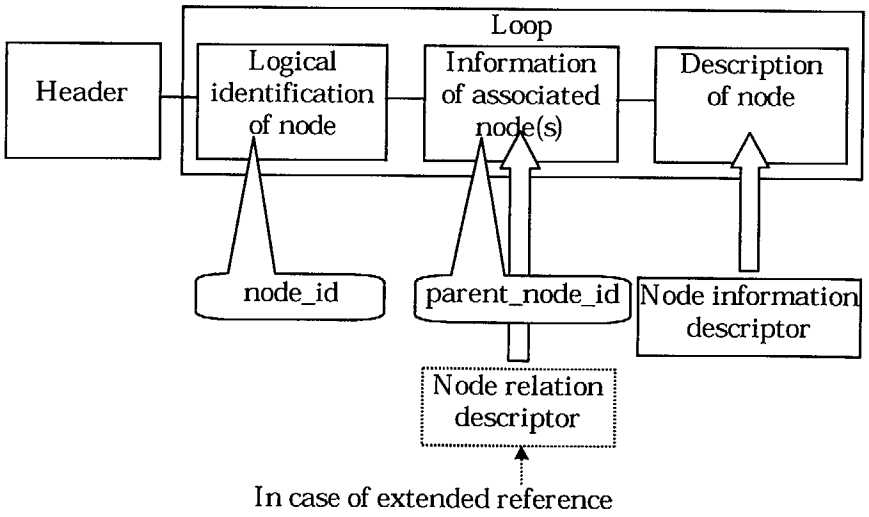


Figure 17 : Structure of ERT

Figure 17 shows the simplified structure of ERT. Relation among events and/or local events can be represented using nodes. In this structure, fields for the logical identification of the node, information of associated node(s) and description of the node are repeated for each node to define node relationship.

Figure 18 shows an example of the Program Index using EIT, LIT and ERT.

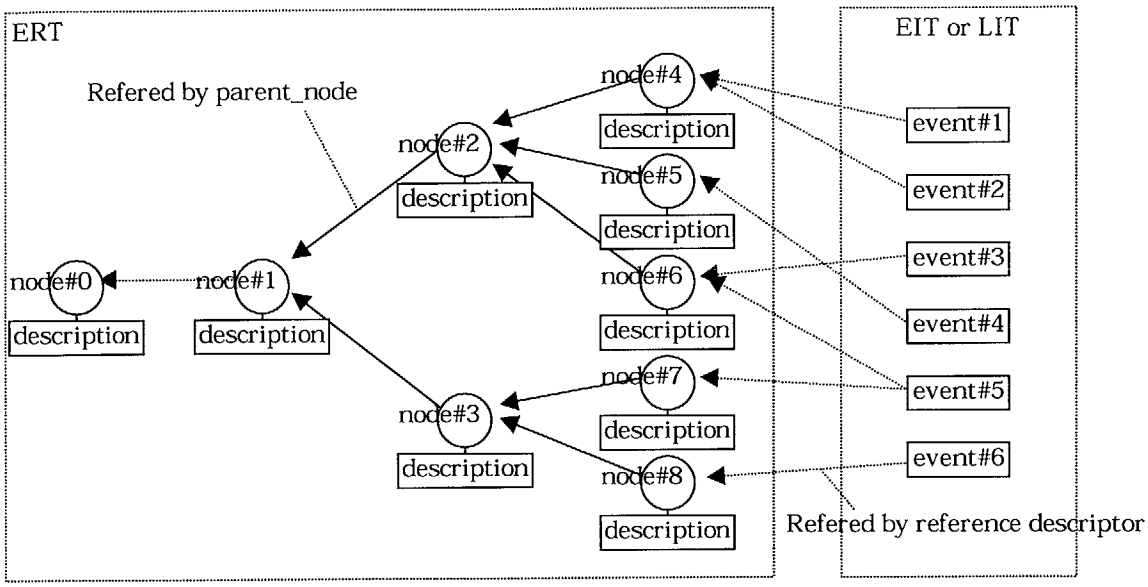


Figure 18 : ERT Tree structure

5.1.4.4 Examples of Index Coding

5.1.4.4.1 Example of Hierarchical Program (News Program)

This section shows an example of LIT and ERT, in the case of a hierarchical news program shown in Figure 19.

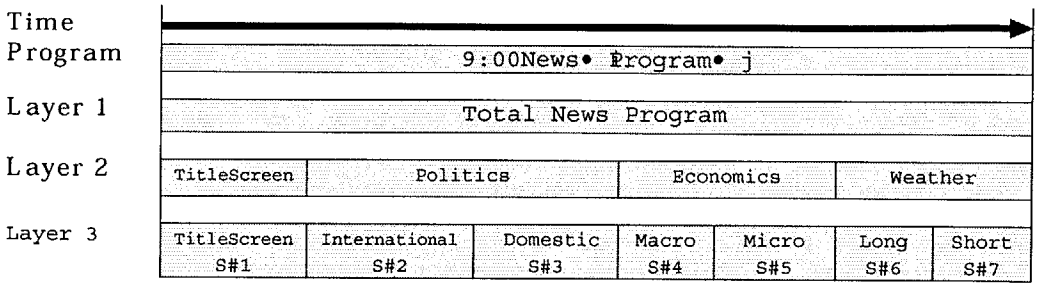


Figure 19 : Model of a hierarchical structured program

Figure 20 shows the data model structure in the ERT for a broadcast program which has the hierarchical structure shown in Figure 19.

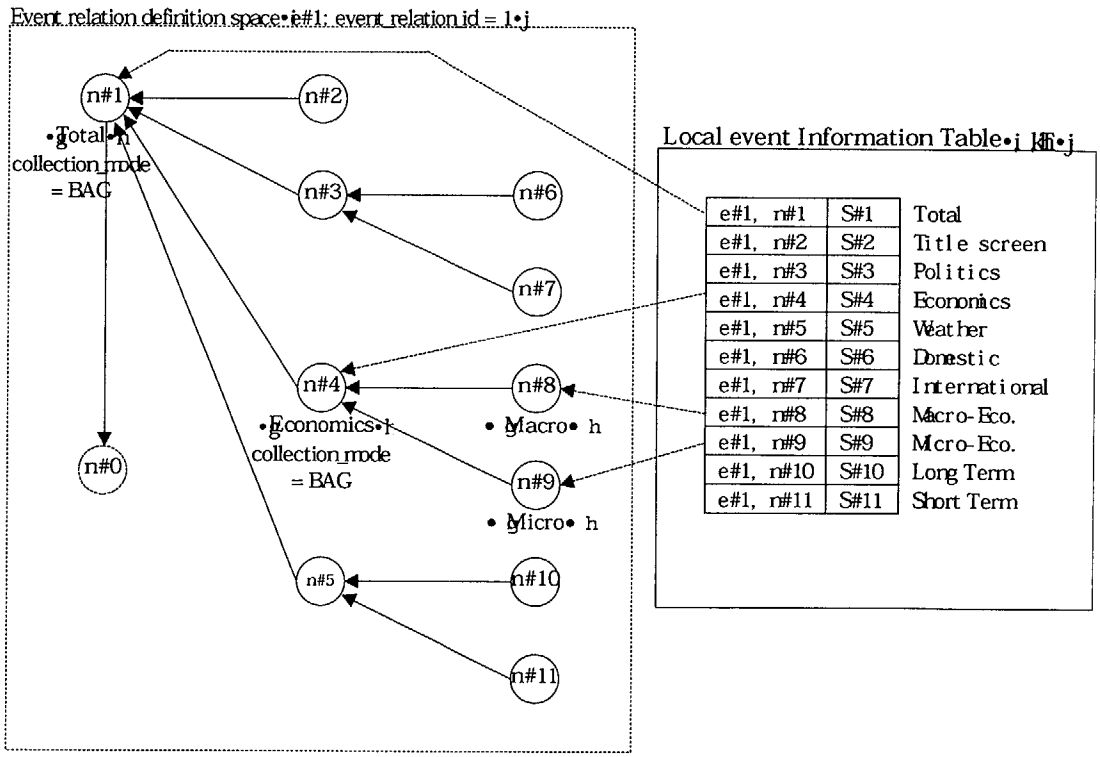


Figure 20 : Hierarchical event model mapping

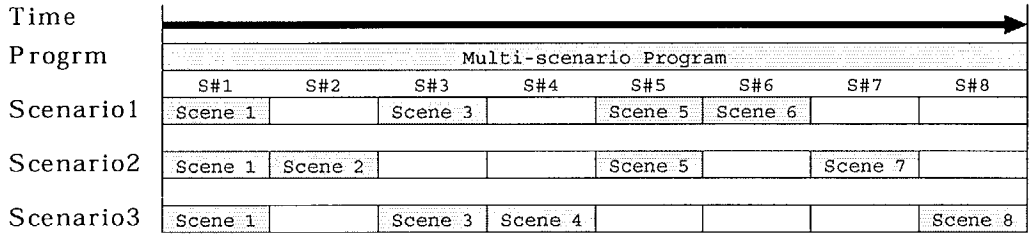
Figure 19 has three layers - Total program, category (title screen, politics, economics, weather) and individual news items in each category. One can see an item in each category. Unlike example 1, a reference from a local event (LIT) is made not only to a leaf node but also to an intermediate node in a tree structure implemented in the ERT. In example 1, if one selects a whole program, the system must search the tree structure; in the example 2, this search is unnecessary.

If a reference is allowed to an intermediate node from a local event, it is necessary to define the scope of the collection_mode in the operation. It is also possible to limit only one reference to an intermediate node from a local event.

5.1.4.4.2 Example of Multi-scenario program

A scenario is defined as a series of program segments (as well as the order of the scenes) that the program producer wants to show. A multi-scenario program is a program in which multiple scenarios (producer's intention) have been defined.

Figure 21 shows a multi-scenario program example. Based on this example, the LIT and ERT tables are described.



- E Local event replay order in each scenario

```
Scenario1 : scene1 -> scene3 -> scene5 -> scene6
Scenario2 : scene1 -> scene2 -> scene5 -> scene7
Scenario3 : scene1 -> scene3 -> scene4 -> scene8
```

Figure 21 : Model of Multi-scenario program

If a program has the multi-scenario structure shown in Figure 21, one can select any scenario to view. Figure 22 shows an example of the event table data model structure for such a program.

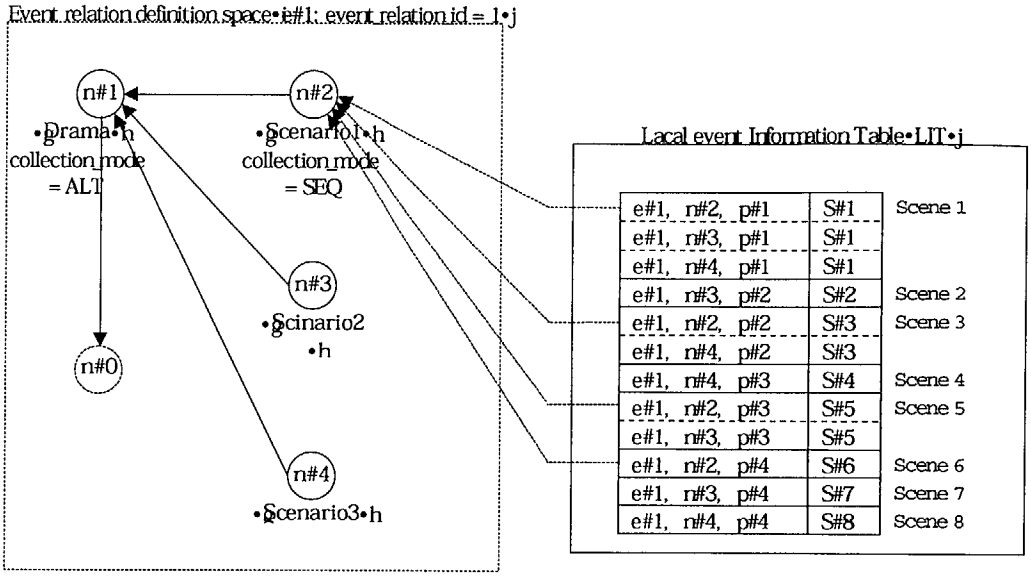


Figure 22 : Event table data model in the multi-scenario structure

5.1.4.5 Specifications

5.1.4.5.1 General Information on Program Index System

5.1.4.5.1.1 Program Index

The Program Index is designed to provide information to assist selection and search of broadcast programs.

The Program Index is provided by EIT (Event Information Table) of program schedule information. EIT defines broadcast programs (events) and describes the index information of the broadcast programs in text.

5.1.4.5.1.2 Program Group Index

The Program Group Index provides grouping information of broadcast programs. It enables, for example, grouping of broadcast programs by the category, grouping of series of programs, grouping of programs with the same contents (rebroadcast), and grouping of recommended programs. This grouping information assists selection and search of broadcast programs.

The Program Group Index is provided by EIT of program scheduling information and ERT (Event Relation Table) which are defined in this chapter. EIT defines broadcast programs and, at the same time, describes grouping information for broadcast programs in plain text or in the program group codes defined by ERT. ERT defines the program group and describes the attribute information in text. ERT can also express relations between program groups.

The Program Group Index can also group segments of a broadcast program. In this case, LIT (explained below in the section about Program Segment Index) is used to identify segments of a broadcast program.

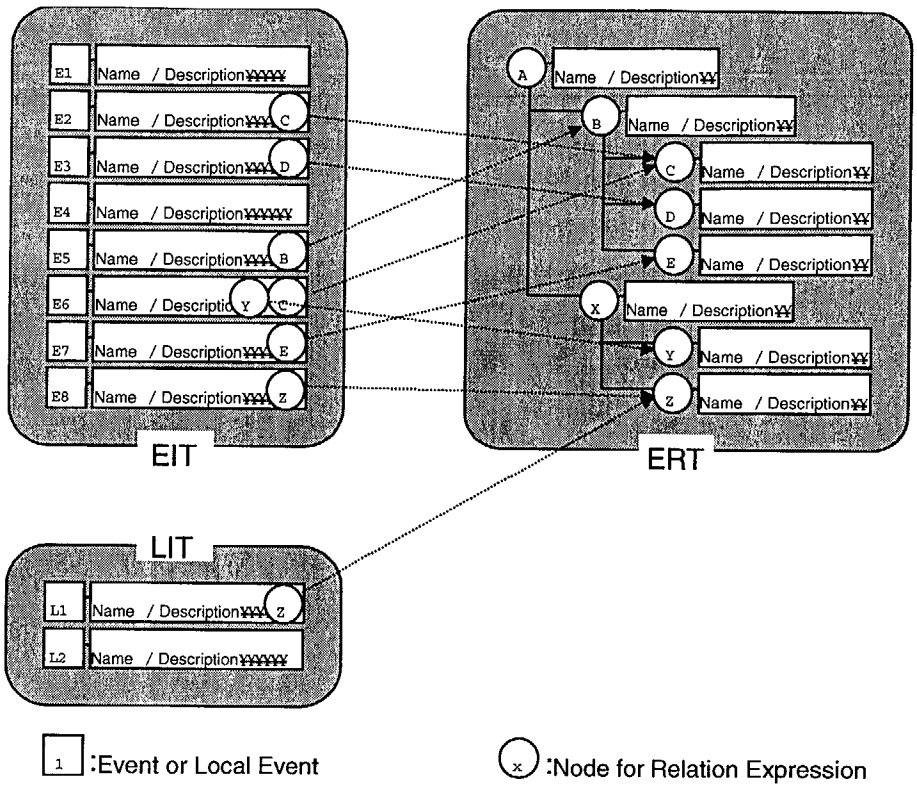


Figure 23 : General Information on Program Group Index

5.1.4.5.1.3 Program Segment Index

The Program Segment Index provides grouping information on events in a program. It helps selection and search of events in a program.

The Program Segment Index is provided by LIT (Local event Information Table), which is defined in this chapter, and ERT. LIT defines a program segment event, and, at the same time, describes in binary codes the grouping information defined by ERT. ERT defines the grouping information of program segment events and describes the grouping information in text.

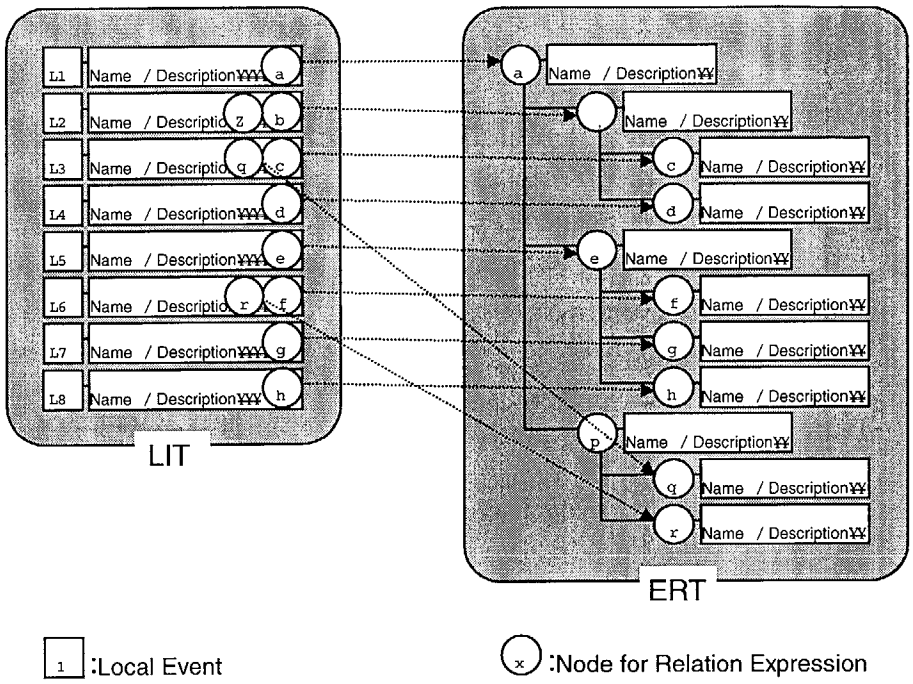


Figure 24 : General Information on Program Segment Index

5.1.4.5.2 Index Encoding System

5.1.4.5.2.1 Local event Information Table (LIT)

LIT shall be encoded using local event information section specified in Table 5-1.

Table 5-1 Local Event Information Section

Syntax	No. of bits	Identifier
local_event_information_section(){		
Table_id	8	uimsbf
Section_syntax_indicator	1	bslbf
Reserved_future_use	1	bslbf
Reserved	2	bslbf
Section_length	12	uimsbf
Event_id	16	uimsbf
Reserved	2	bslbf
Version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
service_id	16	uimsbf
transport_stream_id	16	uimsbf

original_network_id		16	uimsbf
for(i=0;i<N;i++){			
	local_event_id	16	uimsbf
	reserved_future_use	4	bslbf
	descriptors_loop_length	12	uimsbf
	for(j=0;j<descriptors_loop_length;j++){		
	descriptor()		
	}		
}			
CRC_32		32	rpchof
}			

Semantics for the local event information section.

table_id: Indicates the local event information section.

section_syntax_indicator: The section_syntax_indicator is a 1-bit field which shall be set to "1".

section_length: This is a 12-bit field. It indicates the remaining number of bytes immediately following the section_length field up to and including the CRC. The section_length shall not exceed 4093.

event_id: This 16 bit field indicates the identification number (uniquely assigned in an identified service) of the event that the local event information section describes.

version_number: This 5-bit field is the version number of the table. The version_number shall be incremented by 1 when a change in the information carried within the table occurs. When it reaches value 31, it wraps around to 0. When the current_next_indicator is set to "1", then the version_number shall be that of the currently applicable table defined by the table_id and event_id. When the current_next_indicator is set to "0", then the version_number shall be that of the next applicable table defined by the table_id and event_id.

current_next_indicator: This 1-bit indicator, when set to "1" indicates that the table is the currently applicable table. When the bit is set to "0", it indicates that the table sent is not yet applicable and shall be the next table to be valid.

section_number: This 8-bit field gives the number of the section. The section_number of the first section in the table shall be "0x00". The section_number shall be incremented by 1 with each additional section in the local event information table.

last_section_number: This eight bit field indicates the section number for the last section of the table to which the section belongs.

service_id: This 16 bit field indicates the identification number (uniquely assigned in an identified network) of the service to which the event described by the local event information section belongs. The service identifier is equivalent to the program_number in the corresponding program map section.

transport_stream_id: This 16 bit field indicates the identification number (uniquely assigned in an identified network) for the transport stream to which the event described by the local event information section belongs.

original_network_id: This 16 bit field indicates the identification number of the original network to which the event described by the local event information section belongs.

local_event_id: This 16 bit field indicates the local event.

descriptors_loop_length: This 12 bit field specifies the total byte length of the subsequent descriptor.

CRC_32: This 32 bit field indicates the CRC value for the entire section.

5.1.4.5.2.2 Event Relation Table (ERT)

ERT shall be encoded using the event relation sections specified in Table 5-2.

Table 5-2 Event Relation Section

Syntax	No.of bits	Identifier
event_relation_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
event_relation_id	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
information_provider_id	16	uimsbf
relation_type	4	uimsbf
reserved_future_use	4	bslbf
for(i=0;i<N;i++){		
node_id	16	uimsbf
collection_mode	4	uimsbf
reserved_future_use	4	bslbf
parent_node_id	16	uimsbf
reference_number	8	uimsbf
reserved_future_use	4	bslbf
descriptors_loop_length	12	uimsbf
for (j=0;j< descriptors_loop_length;j++)		
descriptor()		
}		
}		
CRC_32	32	rpchof
}		

Semantics for the event relation section:

table_id: Indicates the event relation table.

section_syntax_indicator: The section_syntax_indicator is a 1-bit field which shall be set to "1".

section_length: This is a 12-bit field. It indicates the remaining number of bytes immediately following the section_length field up to and including the CRC. The section_length shall not exceed 4093.

event_relation_id: This 16 bit field indicates the event relation.

version_number: This 5-bit field is the version number of the table. The version_number shall be incremented by 1 when a change in the information carried within the table occurs. When it reaches value 31, it wraps around to 0. When the current_next_indicator is set to "1", then the version_number shall be that of the currently applicable table defined by the table_id and event_relation_id. When the current_next_indicator is set to "0", then the version_number shall be that of the next applicable table defined by the table_id and event_relation_id.

current_next_indicator: This 1-bit indicator, when set to "1" indicates that the table is the currently applicable table. When the bit is set to "0", it indicates that the table sent is not yet applicable and shall be the next table to be valid.

section_number: This 8-bit field gives the number of the section. The section_number of the first section in the table shall be "0x00". The section_number shall be incremented by 1 with each additional section in the event relation table.

last_section_number: This eight bit field indicates the section number for the last section of the table to which the section belongs.

information_provider_id: This 16 bit field identifies the information provider who specifies this event relation.

relation_type: This four bit field expresses the type of relation described by the event relation section. See Table 5-3.

Table 5-3 Relation Type

relation_type value	Description
0x00	Reserved
0x01	Relation for the contents description
0x02	Relation for navigation (Expresses the tree structure to enable display and selection)
0x03-0x0f	Reserved for future use

node_id: This 16 bit field identifies the node used to describe the relations of events and local events. The node identifier "0x0000" is reserved as a special node identifier for the description of event relation sub-table. The node identifier "0xFFFF" is not used.

collection_mode: This two bit field indicates the characteristics of the collection of events, local events, and nodes which refer to this node by the parent node identifier, node relation descriptor, or reference descriptor. See Table 5-4.

Table 5-4 Characteristics of Collection

collection_mode value	Characteristics
0x00	Set
0x01	Sequence
0x02	Choice
0x03	Parallel
0x04-0x0f	Reserved for future use

parent_node_id: This 16 bit field indicates the parent node identifier when another node in the event relation sub-table is referred as a parent. When the parent node identifier is "0xFFFF," the parent node identifier is ignored.

reference_number: This 8 bit field specifies the priority of reference in the node collection which refers to the same node. Smaller value has higher priority for navigation.

descriptors_loop_length: This 12 bit field specifies the total byte length of the subsequent descriptor.

CRC_32: This 32 bit field indicates the CRC value for the entire section.

5.1.4.5.2.3 Index Transmission information Table (ITT)

The ITT describes the information to be used for transmission of Program Index tables.

The ITT is a collection of sub-tables. A sub-table includes correspondence between a single event (program) and its related program index tables, and consists of ITT sections. Table 5-5 shows the ITT sections.

Table 5-5 Index Transmission information Table

Syntax	No. of bits	Identifier
index_transmission_information_section(){		
table_id	**8**	**uimsbf**
section_syntax_indicator	**1**	**bslbf**
reserved_future_use	**1**	**bslbf**
reserved	**2**	**bslbf**
section_length	**12**	**uimsbf**
event_id	**16**	**uimsbf**
reserved	**2**	**bslbf**
version_number	**5**	**uimsbf**
current_next_indicator	**1**	**bslbf**
section_number	**8**	**uimsbf**
last_section_number	**8**	**uimsbf**
reserved_future_use	**4**	**bslbf**
descriptors_loop_length	12	uimsbf
for (j=0;j< descriptors_loop_length;j++)		
descriptor()		
}		
CRC_32	**32**	**rpchof**
}		

Semantics of index transmission information section:

table_id: Indicates the index transmission information section.

section_syntax_indicator: The section_syntax_indicator is a 1-bit field which shall be set to "1".

section_length: This is a 12-bit field. It indicates the remaining number of bytes immediately following the section_length field up to and including the CRC. The section_length shall not exceed 4093.

event_id: A 16 bit value to identify the event (program). The index transmission information section concerns the event (program) specified by event_id.

version_number: This 5-bit field is the version number of the table. The version_number shall be incremented by 1 when a change in the information carried within the table occurs. When it reaches value 31, it wraps around to 0. When the current_next_indicator is set to "1", then the version_number shall be that of the currently applicable table defined by the table_id and event_id. When the current_next_indicator is set to "0", then the version_number shall be that of the next applicable table defined by the table_id and event_id.

current_next_indicator: This 1-bit indicator, when set to "1" indicates that the table is the currently applicable table. When the bit is set to "0", it indicates that the table sent is not yet applicable and shall be the next table to be valid.

section_number: This 8-bit field gives the number of the section. The section_number of the first section in the table shall be "0x00". The section_number shall be incremented by 1 with each additional section in the index transmission information table.

last_section_number: This eight bit field indicates the section number for the last section of the table to which the section belongs.

descriptors_loop_length: This 12 bit field specifies the total byte length of the subsequent descriptor.

CRC_32: This 32 bit field indicates the CRC value for the entire section.

5.1.4.5.2.4 Descriptors

The following descriptors are defined additionally to DVB-SI.

(1) Basic local event descriptor

(2) Reference descriptor

(3) Node relation descriptor

(4) Short node information descriptor

(5) STC reference descriptor

In addition, the following existing DVB descriptors are used.

(6) Short event descriptor

(7) Extended event descriptor

(8) Stuffing descriptor

The following semantics apply to all the descriptors defined in this subclause.

descriptor_tag: The descriptor tag is an 8-bit field which identifies each descriptor. MPEG-2 normative values are described in ISO/IEC 13818-1 [1]. The values of descriptor_tag are defined in Table 5-14.

descriptor_length: The descriptor length is an 8-bit field indicating the total number of bytes of the data portion of the descriptor following the byte defining the value of this field.

5.1.4.5.2.4.1 Basic Local Event Descriptor

The basic local event descriptor used in LIT indicates segmentation information of the local event, such as the start time and the duration. See Table 5-6.

Table 5-6 Basic Local Event Descriptor

Syntax	No.of bits	Identifier
basic_local_event_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
reserved_future_use	4	Bslbf
segmentation_mode	4	uimsbf
segmentation_info_length	8	uimsbf
if(segmentation_mode == 0){		
}		
else if(segmentation_mode == 1){		
reserved_future_use	7	bslbf
start_time_NPT	33	uimsbf
reserved_future_use	7	bslbf
end_time_NPT	33	uimsbf
}		

```
        else if(segmentation _mode < 6){
                start_time                                  24    uimsbf
                duration                                    24    uimsbf
                if(segmentation_info_length == 10){
                        start_time_extension                12    uimsbf
                        reserved_future_use                  4    bslbf
                        duration_extension                  12    uimsbf
                        reserved_future_use                  4    bslbf
                }
        }
        else{
                for( i=0; i< M; i++){
                        reserved                             8    bslbf
                }
        }
        for( i=0; i<N; i++){
                component_tag                                8    uimsbf
        }
}
```

Semantics for the basic local event descriptor:

segmentation_mode: This 4 bit field specifies the encoding method of segmentation specification. (see table 5-7)

Table 5-7 Segmentation mode

Segmentation_mode	Type	Semantics
0x00	None	No relation is specified
0x01	NPT	Normal Play Time
0x02	relative time	Relative time from the beginning of the program
0x03	relative time with compensation	Relative time from the beginning of the program with time compensation by STC reference descriptor.
0x04	Standard time	Standard time
0x05	Standard time with compensation	Standard time of the region with time compensation by STC reference descriptor.
0x6-0xf	reserved	Reserved future use

segmentation_info_length: This 8-bit field indicates the total number of bytes of the segmentation information immediately following this field.

start_time_NPT: This 24 bit field specifies the start time of the local event using the format of NPT.

end_time_NPT: This 24 bit field specifies the end time of the local event using the format of NPT.

start_time: This 24 bit field specifies the start time of the local event with the granularity of more than a second. Using six 4-bit binary-coded decimal numbers (BCD), the time is coded in the order of hour, minute, and second. When no start time is defined (for example, the start time remains undetermined, or it is not shown), all bits in this field must be set at "1."

duration: This 24 bit field specifies the duration of the local event with the granularity of more than a second . Using six 4-bit binary-coded decimal numbers (BCD), the time duration is coded in the order of hour, minute, and second. When no time duration is defined (for example, the time duration remains undetermined, or it is not

shown), all bits in this field must be set at "1." The value for this field must be set to "0" to indicate a point on the time base.

start_time_extension: This 12 bit field specifies the start time of the local event with the granularity of less than a second. Using three 4-bit binary-coded decimal numbers (BCD), the time is coded in milli-second. When no start time is defined, all bits in this field must be set at "1." This field is omitted when no specification is made down to the milli-second level accuracy.

duration_extension: This 12 bit field specifies the duration of the local event with the granularity of less than a second. Using three 4-bit binary-coded decimal numbers (BCD), the time is coded in milli-second. When no time duration is defined, all bits in this field must be set at "1." The value for this field must be set at "0" to indicate a point on the time base. This field is omitted when no specification is made down to the milli-second level accuracy.

component_tag: This 8 bit field identifies the component stream within this local event. The component stream to which the same value of this component tag is assigned in the PMT belongs to this local event. This field could be omitted if all the component streams belong to this local event. This field has the value of "0xff" if none of the component stream belong to this local event.

5.1.4.5.2.4.2 Reference Descriptor

The reference descriptor used in EIT or LIT associates the event or the local event with the event relation sub-table. The reference descriptor refers to the event relation sub-table and decodes the text value from the encoded node identifier. See Table 5-8.

Table 5-8 Reference Descriptor

Syntax	No. of bits	Identifier
reference_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
information_provider_id	16	uimsbf
event_relation_id	16	uimsbf
for (i=0;i<N;i++){		
reference_node_id	16	uimsbf
reference_number	8	uimsbf
last_reference_number	8	uimsbf
}		
}		

Semantics of the reference descriptor:

information_provider_id: This 16 bit field indicates the information provider id of the event relation table.

event_relation_id: This 16 bit field indicates the event relation id of the event relation table.

reference_node_id: This 16 bit field idicates the node which the event or the local event refers.

reference_number: This 8 bit field specifies the reference priority of the reference nodes referred by reference_node_id. In case the referred node indicates the event or the local event itself, the value of this field should be "0x00". In case the referred node indicates the parent node of the event or the local event, the priority should be specified by the value calculated based on the following equation.

reference_number = mod(reference priority order - 1 , 254) + 1

The value "0xff" should be used if no need to specify the priority order.

last_reference_number: This 8 bit field indicates the reference_number for the last reference priority order. The value may be "0xff" if no need for specifying this number. The last_reference_number should not be equal to the reference_number, except for the event or local event of the last reference priority order.

5.1.4.5.2.4.3 Node Relation Descriptor

The node relation descriptor is used to describe the referencing relation of nodes in event relation table. See Table 5-9. In case the referencing is only made to the parent node and the parent node is located in the same event relation identifier, the parent node identifier field of ERT section is used to express the node relation.

Table 5-9 Node Relation Descriptor

Syntax	**No. of bits**	**Identifier**
node_relation_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
reference_type	4	uimsbf
external_reference_flag	1	bslbf
reserved_future_use	3	bslbf
if (external_reference_flag == 1){		
information_provider_id	16	uimsbf
event_relation_id	16	uimsbf
}		
reference_node_id	16	uimsbf
reference_number	8	uimsbf
}		

Semantics of the node relation descriptor:

reference_type: This 4 bit field indicates the reference attribute for the node indicated by reference_node_id. See Table 5-10.

Table 5-10 Reference type

Reference_type value	Description
0x0	Reference to the parent node
0x1 - 0xf	Reserved for future use

external_reference_flag: This 1 bit field specifies if the reference node id is an external reference or not. The value "0" indicates the node referred by the reference node id is located in the same event relation table, while the value "1" indicates the node referred by the reference node id is located in the other event relation table.

information_provider_id: This 16 bit field identifies the information provider id of the event relation table.

event_relation_id: This 16 bit field identifies the event relation id of the event relation table.

reference_node_id: This 16 bit field indicates the node to be referred to.

reference_number: This 8 bit field specifies the reference priority of the reference nodes referred by reference_node_id. The value "0xff" may be used if there is no need to specify the priority order.

5.1.4.5.2.4.4 Short node information descriptor

The short node information descriptor used in event relation table expresses the node name as well as the descriptions on the node definition in the textual format. See Table 5-11.

Table 5-11 Short Node Information Descriptor

Syntax	No. of bits	Identifier
short_node_information_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
ISO_639_language_code	24	bslbf
node_name_length	8	uimsbf
for (i=0;i< node_name_length;i++){		
node_name_char	8	uimsbf
}		
text_length	8	uimsbf
for (i=0;i<text_length;i++){		
text_char	8	uimsbf
}		
}		

Semantics of the short node information descriptor:

ISO_639_language_code: This 24 bit field indicates the language of the subsequent character information field in a form of three alphabetical characters specified by ISO639-2. Each character is encoded in eight bits in accordance with ISO8859-1 and inserted into the 24 bit field in the same order as that of the character code.

node_name_length: This 8 bit field indicates the byte length of the following node name.

node_name_char: The series of character information indicates the node name.

text_length: This eight bit field indicates the byte length of the following node description.

text_char: The sequence of characters provides an explanation of the node.

5.1.4.5.2.4.5 STC reference descriptor

The STC reference descriptor is used to describe the reference between the time specification in LIT and STC. See Table 5-12.

Table 5-12. STC Reference Descriptor

Syntax	No. of bits	Identifier
STC_reference_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
reserved_future_use	3	bslbf

```
        external_event_flag                                  1    bslbf
        STC_reference_mode                                   4    uimsbf
        if (external_event_flag == 1){
                external_event_id                           16    uimsbf
                external_service_id                         16    uimsbf
                external_network_id                         16    uimsbf
        }
        if (STC_reference_mode == 0){
        }
        else if (STC_reference_mode == 1){
                reserved_future_use                          7    bslbf
                NPT_reference                               33    uimsbf
                reserved_future_use                          7    bslbf
                STC_reference                               33    uimsbf
        }
        else if (STC_reference_mode == 3 ||
           STC_reference_mode == 5 ){
                time_reference                              24    uimsbf
                time_reference_extention                    12    uimsbf
                reserved_future_use                         11    bslbf
                STC_reference                               33    uimsbf
        }
        else{
                for( i=0; i<N; i++){
                        reserved                             8    bslbf
                }
        }
}
```

Semantics of STC reference descriptor:

external_event_flag: This 1-bit indicator, when set to "1" indicates that the STC reference descriptor is for the original program provided as another event.

external_event_id: This 16-bits field indicates the event id of the original program in case external_event_flag is set to "1".
external_service_id: This 16-bits field indicates the service id of the original program in case external_event_flag is set to "1".
external_network_id: This 16-bits field indicates the network id of the original program in case external_event_flag is set to "1".
STC_reference_mode: This 4 bit field indicates the type of reference information. This field should be encoded corresponding to the segmentation mode of LIT. See Table 5-13.

Table 5-13 STC reference mode

STC reference mode	Type	Semantics
0x0	None	No relation is specified
0x1	NPT	Reference information for NPT
0x2	reserved	Reserved
0x3	relative time	Reference information for relative time from the beginning of the

		program
0x4	reserved	Reserved
0x5	standard time	Reference information for standard time
0x6-0xf	reserved	Reserved

STC_reference: This field is a 33 bit unsigned integer that indicates the STC value for which the NPT equals the value given in the NPT_Reference field.
NPT_reference: This field is a 33 bit unsigned integer that indicates the NPT value for which the STC equals the value given in the STC_Reference field.
time_reference: This 24 bit field specifies the time reference with the granularity of more than a second for which the STC equals the value given in the STC_reference field. Using six 4-bit binary-coded decimal numbers (BCD), the time is coded in the order of hour, minute, and second.

time_reference_extension: This 12 bit field specifies the time reference with the granularity of less than a second for which the STC equals the value given in the STC_reference field. Using three 4-bit binary-coded decimal numbers (BCD), the time is coded in milli-second. The value "0" is specified in case there is no use for fine granularity.

5.1.4.5.2.4.6 Allocation of the tag value and possible locations of the descriptors

Table 5-14 shows allocation of the tag value and possible locations of the descriptors in the index encoding system.

Table 5-14 Allocation of the tag value and possible locations of the descriptors

Descriptor	Tag Value	EIT	LIT	ERT	ITT
short event descriptor	0x4d	*	*		
extended event descriptor	0x4e	*	*		
basic local event descriptor	0xd0		*		
reference descriptor	0xd1	*	*		
node relation descriptor	0xd2			*	
short node information descriptor	0xd3	*		*	
STC reference descriptor	0xd4				*
stuffing descriptor	0x42	*	*	*	*

5.1.5 Programme Identifiers and Locators

5.1.5.1 Universal Programme Identification – UPI

Content is uniquely identified by a so-called UPI (Universal Programme Identifier). We propose the following URN based format (see RFC 2141) for the UPI:

- URN: Uniform Resource Name

A persistent and location independent handle to a UPI.

Syntax: <NID>.<NSS>

NID: namespace ID

NSS: Namespace Specific String

- UPI: Universal Programme Identifier

  Syntax: <URN>

  <NID> .<NSS>

  < Set UPI> .<SNID> .<SNSS>

  .<sub namespace ID> .<subNamespace Specific String>

- Set UPI: Universal Programme Identifier - Set

  The parent of a set of UPI children, it may resolve into further Set UPIs.

- Leaf UPI: Universal Programme Identifier – Leaf

  Contains either the information for one broadcast (a bounded response) or a UPI-Unbounded response. A UPI-Unbounded also includes a time, which is then scheduled into the STB for the next resolution. The next resolution may resolve into either another UPI-Unbounded reply or information about a specific broadcast.

- UPI-Unbounded

  A Leaf UPI for a broadcast, along with a time to schedule the next resolution of itself.

As an example, valid UPIs (as a URN namespace) could read:

"urn:upi:bbc:12345"

"urn:upi:abc:12345"

"urn:upi:smpte:12345"

where the upi is a registering authority under a number of SNID – resolving authorities will be declared. The "SNID-bbc" (or SNID-abc or SNID-smpte) represents the resolving authority that is responsible for uniquely assigning "SNSS" numbers (e.g. UPID, UMID, UPN) to content within this "SNID-bbc" namespace. This general representation allows to put a link to the corresponding programme, on a web page. Using UPI as a URN namespace allows to refer to the same piece of content either in a broadcast or in an Internet environment.

5.1.5.2 Uniform Resource Locator – URL

The URL is protocol dependent. The following URL formats cover the requirements expressed at the time of producing this DAVIC specification.

5.1.5.2.1 URL for DVB based networks

It was decided to adopt the following DVB extended URL

dvb://<original_network_id>.[<transport_stream_id>][.<service_id>[.<component_tag>]

[;<event_id>]@<start_time>D<duration>][/<...>]

where <start_time> is denoted as yyyymmddThhmm (according to ISO 8601), and duration as hhmm. So a 90 minute BBC 1 programme at 8.15 pm on Christmas eve would be referred to as

dvb://bbc..bbc1;1a639f@19981224T2015D0130

The time and date are given in UTC (Universal Time, Co-ordinated), as this is the format used in the DVB Service Information tables. This avoids problems caused by different time-zones. The use of a textual format for the original network and service Ids has been anticipated.

5.1.5.2.2 URL for IP based networks

The TV-Anywhere concept implies that broadcast contents might also be accessible through the Internet. Therefore, IP URLs can also be the result of the UPI resolving process, for example:

http://www.bbc.co.uk/sport/olympics.http (download example)

http://www.bbc.co.uk/sport/olympics .sdp(stream example)

5.1.5.3 Resolution process

Functionality and associated requirements

1. The UPI is defined by a content provider, service provider or any other third party which wishes to offer a service by resolving this UPI into other UPIs and/or eventually URLs. Resolving may also include e.g. the review of the copyright, transaction and security status at the time of content acquisition.
2. Resolving consists of translating UPIs into unequivocal URLs from which the content will be retrieved. Once content as identified by a UPI has been selected, resolving should occur automatically until the user request is fulfilled. At least one URL has to be provided when the content is available. The content may be available from different locations identified by separate URLs. This variety may provide an advantage in improving the quality of service and resolving availability conflicts following rules to be defined.
3. The UPI and complementary information (e.g. from the SI or from the STU configuration look-up table) may be required to allow the user to resume a transaction and to receive the content when available.
4. The same UPI mechanism will apply for flexible management of local storage resources. Management of the content should allow local stored information to be fully considered in the resolving process. The local storage content manager will play the role of the resolving entity.
5. Some attractors can be attached to the UPI to maintain an access to basic information about the UPI related content. Attractors may also accompany the UPI or URL to facilitate the selection process. Generally, attractors will contain the information on which the user will base his selection.

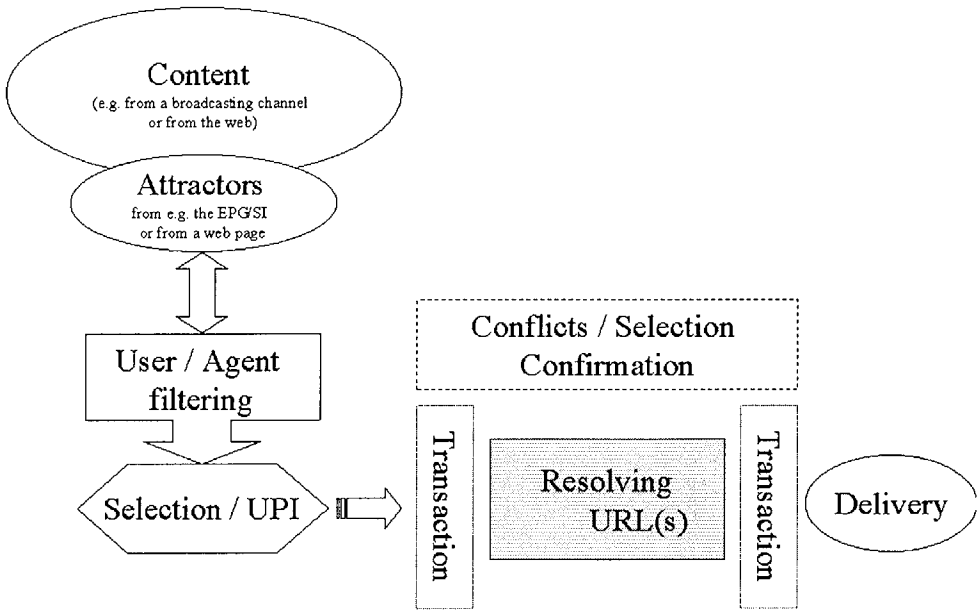


Figure 25 : Search, selection, and fulfilment mechanisms

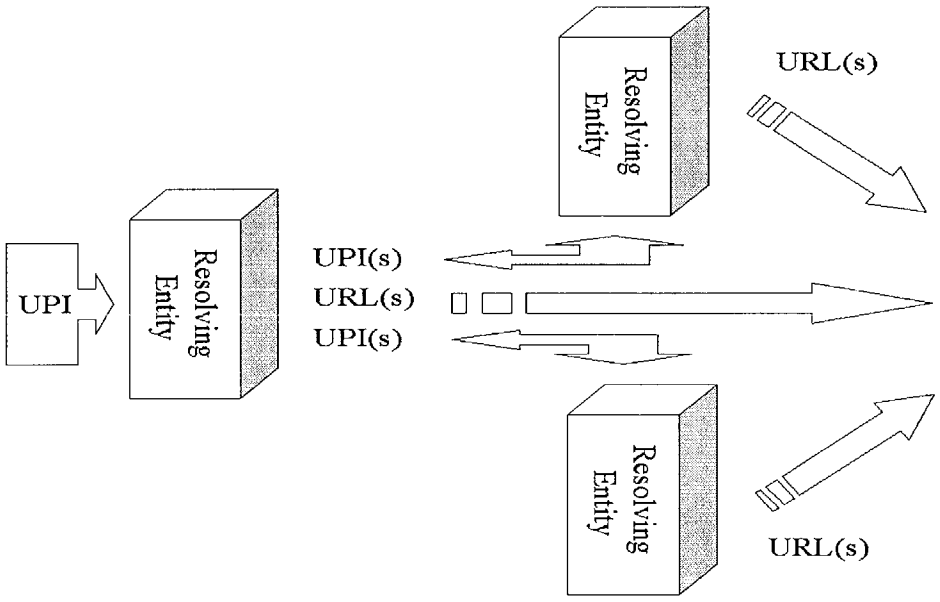


Figure 26 : Example of UPI based resolving mechanism

The resolving entity for a UPI will provide a mechanism by which the UPI can be resolved automatically into other UPIs or into a physical locator which specifies the time and place of broadcast. In some cases this process may require several steps. A UPI can be collective and decompose into subsequent UPIs (e.g. a "series" UPI expanded into "episode" UPIs). Once content corresponding to a collective UPI has been selected for recording, all related content will be captured automatically.

For each content item to be captured at least one physical locator is eventually required. In the case where multiple locators exist (e.g. repeated transmissions of the same content, where one locator or another may be

used) the locator to be used may be chosen according to criteria such as time, cost, quality, or conflict with other content recording.

While unique UPIs can reference complex programme structures based on relational links between content items, the search and filtering tools may deliver to the capture or fulfilment process a single ID corresponding to any point in the decomposition chain.

Once given to the capture process, fulfillment will result in the capture of all content items hierarchically decomposed from that point.

If a user or agent wishes to move up the decomposition chain – say where a user is attracted to capture a series having seen a single episode – mechanisms will need to be provided to allow users to derive any collective UPI from the resolving entity. Alternatively, the service provider may wish to insert explicit links to the series or any other related content into the content itself, using the same mechanism described for promos and trailers.

Implementing the DAVIC UPI and URL concept can be imagined in many different ways. The resolving function is addressing a number of key features like the definition of relationship between different contents.

In order to ensure interoperability, the following resolving scheme has been adopted:

1. A UPI is selected either by a user or an agent using parameters defined by the user. The selection is done via a remote control and a user interface on the television, or by using a computer to set up the required parameters.
2. The STB decodes and then extracts the Resolving Authority (RA) from the UPI. The RA is located within the SNID part of the NSS.
3. The STB locates the RA, using one of the following methods
    1. A lookup in a mapping table located on the STB to locate the required entry.
    2. Via the internet using a RA Discovery Server to search for and return the required entry.
    3. A cache based system using both the STB mapping table and the RA Discovery Server

        (i) Look in STB mapping table, does the RA have an in-date entry

        (ii) YES: Go to the entry

        (iii) NO: Jump to the RA discovery server, pointed to by a permanent persistent entry in the mapping table.

        (iv) Retrieve, from the discovery server, the entry for the RA.

        (v) The entry is located depending upon which RA the STB has access too. e.g. Subscription channels

        (vi) Put this entry in the STB's mapping table for later use.

        (vii) Go to the entry.
4. Once the STB has the RA location, it is contacted either by the internet or by using the broadcast stream information
    - Via the internet,

      The STB will query the RA about the UPI, the RA will return a Set UPI which will be resolved to give its Leaf UPI children.
    - Using broadcast stream information

      The STB, will look at any cached stream information for a match on the UPI for a handle to a Set UPI. If the handle is not found in the cache, the broadcast stream is scanned as it arrives for a handle to a Set UPI.
5. Now the STB has the Leaf UPI, it resolves the Leaf UPIs into one or more URLs
6. The URLs resolved from the Leaf UPIs, contain time and channel information, these point the STB to the relevant signal.

7. Should a scheduling change occur, the STB will be notified (flagged). This will cause a new resolution of the UPI Leaf.
8. The content is broadcast, the STB Leaf UPI's URL matches the broadcast information and the content is then captured.

5.1.6 Metadata for TV Anytime and TV Anywhere

5.1.6.1 Introduction

5.1.6.1.1 Definitions

Metadata is generally defined as "data about data". The most visible part of TV-Anytime and TV-Anywhere metadata is the attractors which are used e.g. in electronic programme guide (Service Information – SI), or in web pages, to describe content and incite the end-user to access it.

Practically, metadata is a fully integrated system which can be divided into:

- the metadata descriptors;
- the metadata schema;
- the metadata encoding language;
- the metadata packaging and transport mechanisms;
- the metadata tools e.g. parsing and authoring.

The DAVIC 1.5 metadata specification aims at enlarging the notion of service information:

- the descriptors are clustered into categories and classes (attractors, descriptors, identifiers, locators, security, etc.) necessary to describe, access and manage contents.
- The Universal Programme Identifier is a specific descriptor that plays a key role in ensuring access to the content requested by a user or an agent. It also links pieces of related content. Its functionality and associated requirements are described in the following sections.
- the metadata schema is compatible with the Dublin Core. Filtering masks can be used to extract metadata information and adapt it to the needs of different content types.
- Authoring tools can be developed to help in generating the corresponding metadata.
- The metadata encoding language is necessary to represent the metadata structure and information using a predefined semantics and grammar. XML is considered as a good candidate for future specification (using XML DTDs or RDF).
- This metadata description stream can be formatted and transported (see Section 3.1.6.3.1).. Different tools can be developed to author, analyse and process the metadata information. Different applications will require a different process but two frameworks can be clearly identified whether the information will be automatically processed (e.g. by intelligent agents) or not.

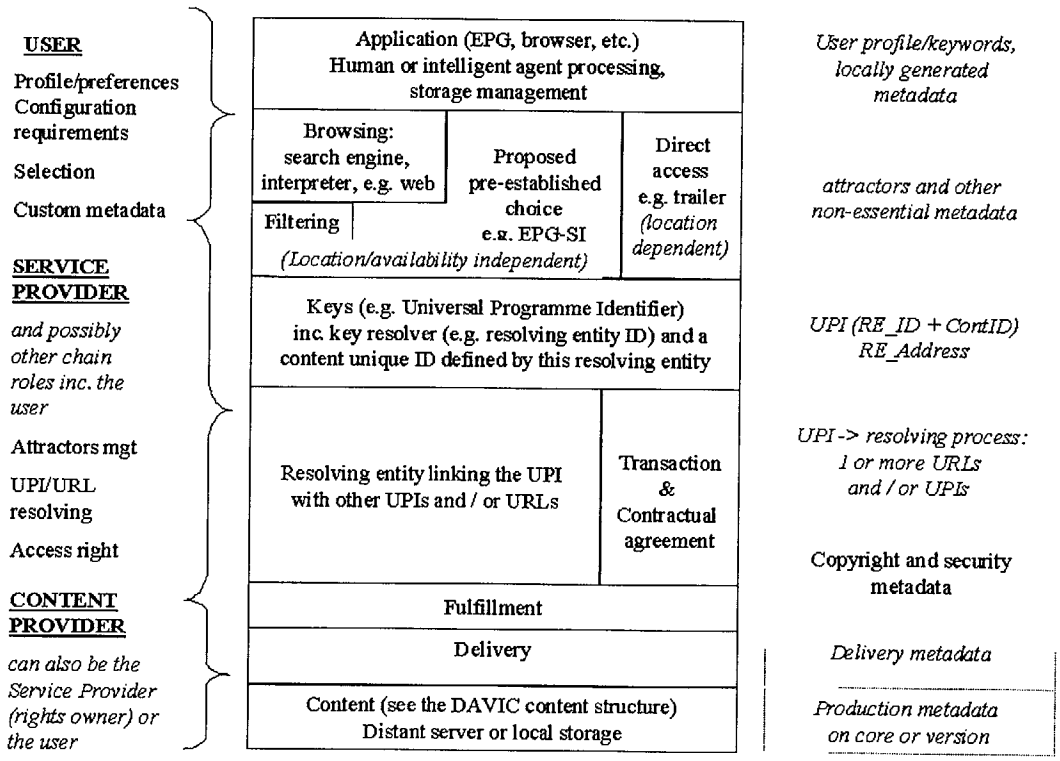
DAVIC has already specified a metadata system for professional exchange of content between content providers and service providers. The UPI identification system allows to maintain backward compatibility with the content structure defined in DAVIC 1.3. The UPI mechanisms can also be used to extend the structure and establish a relationship between a piece of content and its associated segments and / or objects. Only some of the descriptors can currently be shared between production and delivery. The use of segments and objects will probably require further harmonisation work in order to allow an increasing amount of information to be generated at the production stage for further use in distribution.

In the TV-Anytime / TV-Anywhere consumer domain, metadata will ease browsing/searching, selection, access, security, transaction, copyright and local storage management processes. Different roles in the delivery chain will be involved, e.g. in producing market/service tailored attractors, or managing evolving copyright information. Related metadata will therefore be generated and delivered by different sources, in different formats through different delivery media. The different descriptors will be "distributed" along the TV-Anytime / TV-Anywhere value chains. DAVIC aims at specifying a metadata system allowing the highest level of interoperability. This is why the metadata schema was developed for being compatible with the Dublin Core which is gaining growing interest also in the broadcast community.

The metadata system will evolve in time in a backward compatible manner. Metadata versioning will be handled by defining an appropriate namespace.

5.1.6.1.2 Metadata System Reference Model

DAVIC has developed the metadata reference model shown in Figure 27. It highlights the central role of the Universal Programme Identifier (UPI) and associated resolving processes into subsequent UPIs and/or eventually URLs. Different TV-Anytime and TV-Anywhere content access scenarios like direct advertising, electronic programme guide, Internet browsing have been considered supporting human or agent initiated searching and parsing mechanisms.



NOTE change Unique UPI into Universal

Figure 27 : DAVIC Metadata system reference model

Metadata is generally used to help the user to select content (from a selection or as a result of a search) and to manage information about locally stored content.

Each programme is uniquely identified by a Universal Programme Identifier. This identifier will be used as a reference during the fulfilment process until a URL is resolved for content delivery. In this framework, the content material may or may not be immediately available, still allowing the consumer to access it independently of its location in time (the past corresponding to the material preliminary recorded) and space.

UPIs are also used to link related contents.

5.1.6.1.3 Metadata reference flows

This section identifies how metadata can be transmitted in a TVAnytime / TV-Anywhere environment and how the control of that transmission could work.

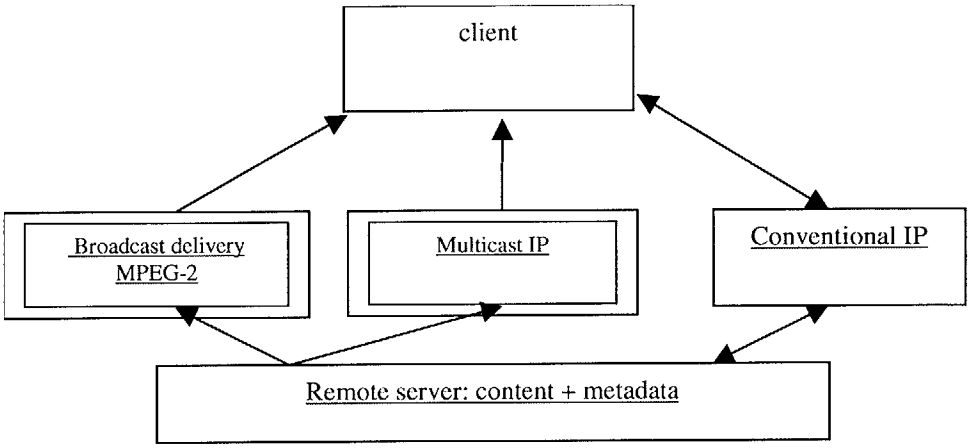


Figure 28 TV-Anytime and TV-Anwhere content and metadata flows

Figure 28 shows three paths for metadata from a server to a client. These paths correspond to delivery via MPEG-2 broadcast, uni-directional IP multicast and conventional bi-directional IP.

5.1.6.1.3.1 MPEG-2 broadcast

In the MPEG-2 broadcast case, a set of metadata is output from a server and packaged for efficient transmission in MPEG-2 based networks. This metadata is used by the client to browse and select content.

5.1.6.1.3.2 Conventional bi-directional IP

The model is totally different for conventional bi-directional IP networks. In this case, metadata is tailored according to the client requests along the browsing phase until the content desired has been selected for delivery.

5.1.6.1.3.3 Uni-directional IP Multicast

For uni-directional IP multicast, the model is very similar to MPEG-2 broadcast. Metadata is basically duplicated from what is available in the broadcast channel. Browsing and selection are based on the same process on the client side.

5.1.6.1.4 Content Structure

Another important element of a metadata description consists of the possibility to link content instances together, each content instance of a given content type possibly having its own metadata description.

The DAVIC structure of content is the foundation on which the UPI links are build.

It is defined in DAVIC 1.3 as a hierarchy of content packages, items, and item elements that would now be extended to include item element segments/objects to which descriptors are attached. This structure is object orientated and encompasses the notion of object modeling inheritance which directly impacts on the quantity of metadata information necessary to fully describe the related content instances.

5.1.6.1.5 Description of the relationship between content elements

5.1.6.1.5.1 Relationship between different related contents

Figure 28 shows how content type can be associated using the DAVIC metadata structure:

- Programme group, e.g. a series, can be represented as the highest hierarchical level, i.e. the content package
- Different programme, e.g. episodes or trailers, can be represented as content items.
- Each programme is made or audio and/or video and /or data, represneted and content item elements

Each CP, CI or CIE has its own Universal Programme Identifier (UPI). Parent/children links will be established by listing the corresponding UPIs.

5.1.6.1.5.2 Segments and object: specific cases

Each progarmme can also be described as segments or objects (see Figure 29). Nevertheless, segments and objects are not identified by separate UPIs. Segments and objects are defined as an extension of the programme UPI.

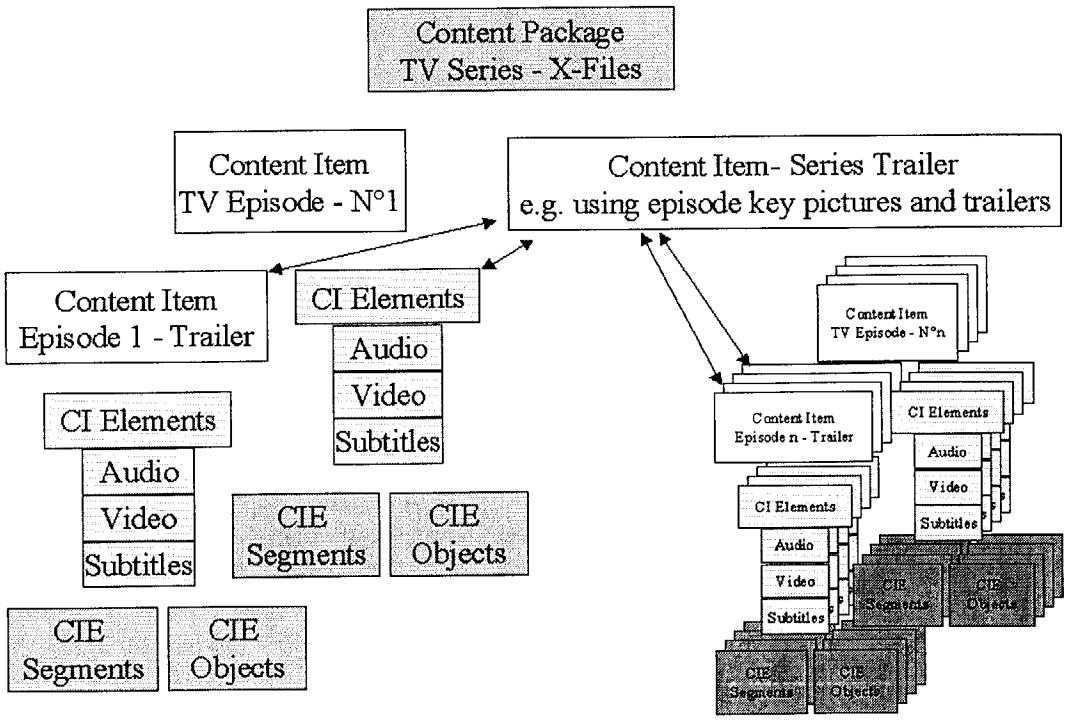


Figure 29 : Example illustrating the relationship between the CP, CI, CIE and CIE Segments / Objects and associated metadata

5.1.6.2 Metadata Schema

5.1.6.2.1 Identification of the metadata specification

The DAVIC core metadata system will cohabit with other metadata systems developed for other application domains. It will therefore be necessary to clearly identify the DAVIC metadata system to ensure that the client use the appropriate system to process incoming metadata.

Identification mechanisms should support automatic or human processing.

One solution could consist of referring to the specification through a specific URL:

Example: DAVIC Metadata Structure Version 1.40

*<xml:namespace name="*http://www.davic.org/*Davic TV-Any/metadata1.40>*

If XML were used as the encoding language (under consideration for future specification work), the DAVIC metadata system could be declared under the DAVIC namespace. This would allow metadata to be properly carried throughout any delivery environment supporting XML.

5.1.6.2.2 Descriptors

The term "descriptors" encompasses a number of elements. They have been defined taking into account the requirements for TV-Anytime and TV-Anywhere:

- Identifiers (linked to the locators)
- Attractors and descriptors
- Access and Security
- Segmentation
- Locally (user or machine) generated extensions

Each class correspond to specific processing stages.

5.1.6.2.2.1 Intrisic , version, essential and non-essential metadata

When content is created, it can be assigned metadata by the "Content provider" according to the production content. This information does not change regardless how the content is delivered. This is considered to be "intrisic" metadata. Intrinsic metadata such as official registration numbers can be used for copy management and protection e.g. for watermarking.

When content is delivered it can be assigned additional metadata by the "Service provider" according to the particular outlet to which, and transmission delivery means by which, it is targeted. This is considered to be "version/instantiation" metadata. As rights can have been transferred or granted under specific and restrictive conditions, complementary registration numbers will be attributed superseding the intrinsic registration numbers. These new registration information will also be usable for e.g. copy management and protection.

Taken together, the intrinsic/core and version metadata provide the client (human or agent) with a range of information which can be used for different applications.

Some descriptors are essential to the description of the content while others are optional (Table 5-15, Table 5-16).

Table 5-15: Examples of "core" metadata

Element:	*Description:*	*DAVIC Status:*
UID	A unique identifier of the content officially registered of locally define	Essential
Human label	A brief textual description of this content, which is suitable for it to be identified by a human being (sometimes called "name",).	Optional
Descriptors/ Attractors	A list of descriptors assigned to the whole content, in order to make it attractive to a consumer of the content (for example, a list of keywords, a content type classification,).	Optional
MPEG7 attributes	For compatibility pruposes	Optional
Duration	The (original) length of time for a linear component.	Optional
Technical format	The (original) technical format specifiers.	Optional
Copyright	A statement of the ownership rights of this content	Essential

Table 5-16: Examples of "version" metadata

Element:	*Description:*	*DAVIC Status:*
UPI	Unique Programme Identifier used by a resolving entity to link interrelated pieces of contents of content elemenst	Essential
ContId for "parent"	A unique identifier (machine readable) of other content which is a super-set of this content (for example, the parent might be a series of programmes of which this content is a single programme).	Essential (if it applies)
List of ContIds for "children"	A list of unique identifiers (machine readable) of other content which makes up this content. For linear content contributions, there also needs to be a definition of the order of the contributions.	Essential (if it applies)
URL	A unique resource locator to identify where the content can be found when it is delivered.	Essential
Date	The date on which this version is delivered	Essential
Time	The time at which this version is delivered	Essential
Duration	The length of time for a linear component.	Desirable
Format	The technical format in which this version is delivered	Optional
Attractors / Descriptors	A list of descriptors tailor-made and assigned to this version, in order to make it attractive to a consumer of the content (for example, a list of keywords, a content type classification,).	Essential
Copyright	A statement of the rights ownership of this version	Essential
Security protection	A security stamp for closed user groups (public/private consumption, visible/invisible level)	Optional
Transaction	Elements of information exchanged during a transaction	Optional

5.1.6.2.3 Metadata schema

The DAVIC metadata schema (see Annex A) is a tree structure compatible with the Dublin Core categorisation scheme (See Table 5-17) developed as a common set of metadata classes for interoperability.

Table 5-17: Dublin core categorisation and classification

Content	Type
	Title
	Subject, keywords
	Description
	Source
	Relation
	Coverage
IPR	Creator
	Publisher
	Contributor
	Rights
Instantiation	Date, time
	Language
	Format
	Identifier

"Content", "IPR" and "Instantiation" are categories under which the DC metadata classes are grouped. Metadata processing occurs only on the DC classes.

The DAVIC metadata schema is itself a common trunk for interoperability. The filtering mask mechanism described in Section 5.1.6.4 allows application specific multi-level extensions to the core schema.

Application schema extensions should also include a version identification that will come e.g. as an extension to the core metadata version number.

5.1.6.2.4 Metadata tailoring/filtering

If it is necessary to have a common reference which is the DAVIC metadata schema, each application or content type does not need to use all the information that it contains.

The extraction of application/content specific metadata information can be performed using a "extraction mask". This means that the bit pattern associated to each node of the metadata schema binary image produced by the filtering mask is kept unchanged. It can nevertheless be extended for specific needs (see Section 5.1.6.4)

Access restriction can apply to various part of this information which can be made invisible or private as shown in Figure 30. Metadata can also be protected against modification or against deletion.

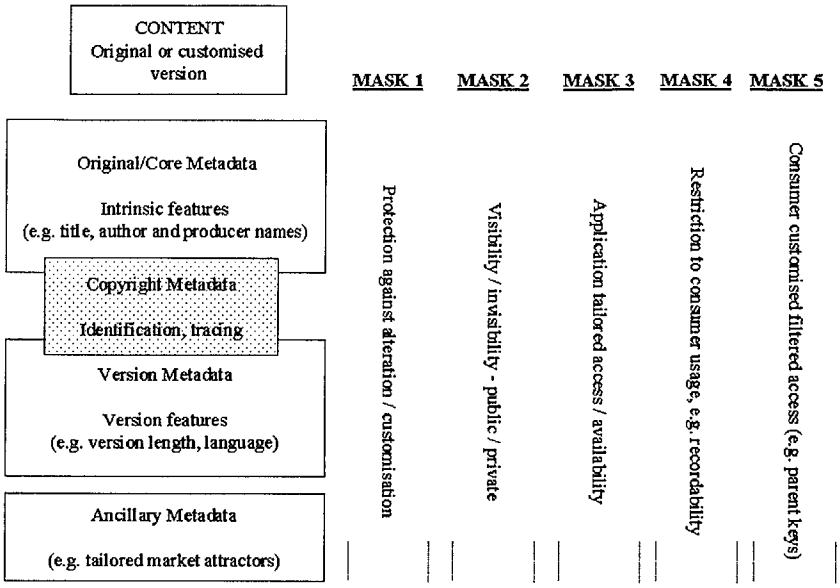


Figure 30 : Metadata access restriction and filtering layers

5.1.6.2.5 Attractors

The attractors contain information that will "attract" the consumer and incite him to access the content they are associated to.

5.1.6.2.6 Locators

Locators will give the consumer the appropriate co-ordinates to access or download the requested content.

5.1.6.2.7 Access & Protection

The access, protection and security information is telling the consumer about the rights of the content and related access conditions. It also includes the information to be exchanged to complete a transaction.

It is acknowledged that "security" is not the most appropriate definition that should apply in this context. Security must primarily be understood as a means to protect contents (rights) and its associated metadata (e.g. containing the copyright information).

For selling and purchasing contents "content provider, service provider and consumer", it is needed to establish a contract between the provider and the consumer.. The Copyright Information defined in DAVIC 1.4 include the CID,CNO,PID,PNO and CTP. The transmission function is retained only in CP and SP, then a sophisticated user having a transmission function is categorized into SP.

- *User responsibility*

The newly defined service consumer who may have an Internet server should be responsible for managing copyrighted contents in line with the requirements under the WIPO's treaty (World Intellectual Property Organisation - Geneva '96), in secure condition. Therefore, a definition for the digital signature should be defined in the Metadata schema relating to CAK of copyright information.

It might be recommended to download the user authorisation status onto the provider site to allow the end-user to download or playback the requested content. This information is defined as part of the copyright and transaction descriptors.

- *CP,SP responsibility*

  Providers should be responsible for autorizing a cleint to access their services. Digital signature is an effective way to certify the PID and PNO.

Licensing

Considering ordinary multimedia contents or new services (in particular using local storage), licensing processes well suit the to smooth accomplishment of contract. Each creation element constituting multimedia or edited content has its own use condition defined by the provider and also from the classical author. The contract and agreement for use of the edited content need detailed contract processes. The licensing metadata information often vary for PID and PNO in contrast to the copyright description of the fundamental attribute such as title, author name, genre etc., referred by CID and CNO. The latter is referred by accessing a database, but the former is maintained in each digital object as licensing metadata instead of some data in a database. The licensing process also involves digital signature together with CAK. In this case, the consumer digital signature is defined as part of thecontent metadata, enhancing the copyright information instead of setting a specificcontract document.

Availability

Provider/author defined use conditions can be translated into a simple CTP through the licensing process. The use condition offered is represented in the content specific in the metadata schema extracted from the DAVIC metadata schema and instantiated with the appropriate values. Resulting licensing and agreement information can eventually be edited as licensing metadata from which the CTP is constructed.

Promotion scheme

Promotion of content is important and requires appropriate protection mechanism for its secure delivery. Specific metadata categories have introduced in the DAVIC schema allow such a promotion in a DAVIC interoperable secure environment . Promotion will have to be done using clear understandable and unalterable information associated to well defined use conditions. The relationship between security and promotion will have to be maintained.

Protection scheme

It is expected that the evolution of technology and services will change the current consumption habits. A wider offer will be available but each content iteme will not be acquired particularly if it has to be purchased. This means that the end-user will be .e.g. willing to try a product before deciding to acquire it. This gives another dimension to copy management and protection. Even if no clear requirement has been expressed yet, it is foreseen that this will require more detailed information, and associated descriptors, on the access conditions and content status in particular for search and trial prior to purchase.

Tamper resistant mechanisms against unauthorized alteration are recommended. In order for metadata and contents to be securely transmitted and used, security tool should be invoked partly by specially notified

security metadata. Other security function may also be ready. Normal keys will be applied to the latter case, and, in the former case a special kind of key will be provided inpependently for each service. Some security metadata shall therefore be in special metadata schema category to allow easy encryption independently from other the possible encryption of other metadata classes. Watermarking may be applied to both but only This special security metadata may be hidden . Other metadata should be visible for most of the providers and the consumers for proper content use even if if watermarking could be applied because of detecting alteration

These security part of the DAVIC metadata schema shall to the largest possible extent be interoperable with other security (e.g. MPEG-4 IPMP) and metadata specifications (e.g. MPEG7).

5.1.6.2.8 Segments and objects

Segmenting is used to navigate within monomedia streams which can also be described as non-linear consultation. Objects are the different elements constituting a piece of content (including segments).

Segments and objects are defined in the DAVIC metadata schema as specific content types to which metadata can be attributed.

Segments and objects do not need to be separately identified. UPI indexing is sufficient, the UPI being the global universal identifier of the overall content (segment – object container).

5.1.6.2.9 Locally generated Extensions

The forthcoming extended capacity of future client installations will allow to store an increasing quantity of contents. The end-user will likely look for applications to help him in managing this information.

Some content related metadata will be an important source of information that the user may wish to use in organising its own database of contents. He may also desire to extend the DAVIC core metadata schema to create complementary categories.

The adaptation of the mechanisms described in this specification for the core metadata schema extension by the service provider is under consideration for future work specification.

5.1.6.3 Metadata Encoding

This chapter specifies the coding syntax for the metadata instance of the DAVIC core schema. The metadata instance has the format of the textual notation. Each node recognized as the descriptors in the DAVIC core schema hierarchy will be assigned a name of unix file system naming syntax. Syntax of the node name represents the hierarchical relationship between the nodes. The coding method allows the user to add additional nodes to the leaf node of the core hierarchy and also allows to add attribute name and value pairs to each node.

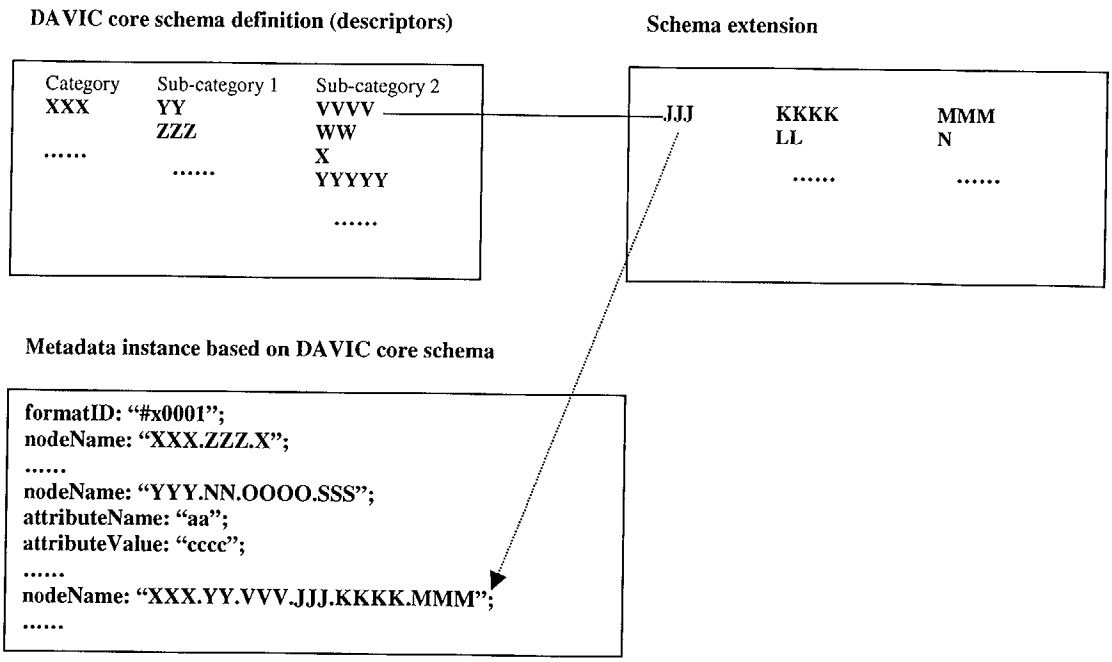


Figure 31 : Metadata Coding for DAVIC Core Schema

5.1.6.3.1 Coding Syntax

Syntax Description

Character set:

Characters are based on the standard ISO10646 character encoding scheme or Unicode. The character set used for metadata broadcasting system is predetermined by the service provider.

Comments are expressed by the text within the /* .... */ code.

The metadata has the following sequence;

FormatID, white space,
Set of (
    NodeName, white space,
    Set of (
        AttributeName, white space,
        AttributeValue, white space
)
)

FormatID is an unique identifier for the encoding syntax.

FormatID is specified by the following sequence;

Reserved word (**formatID**) and immediately followed by a colon(:), a space or a tab, a quotation mark("), hexadecimal integer preceded by a pond sign and the literal string x (#x), a quotation mark(") and immediately followed by a semicolon(;).

The value **#x0001** is reserved for the encoding syntax of metadata instance of the DAVIC core schema.

NodeName expresses the node name of the hierarchical structured nodes corresponds to the metadata descriptor defined in the DAVIC core schema. **NodeName** is specified by the following sequence;

Reserved word (**nodeName**) and immediately followed by a colon(:), a space or a tab, a quotation mark("), text string name of the node in the unix file system naming syntax, a quotation mark(") and immediately followed by a semicolon(;).

NodeName which does not correspond to the node in the DAVIC core schema is allowed to be specified. In this case, the interpreter has to check that the **NodeName** expresses the node which is located under the leaf node of the core schema.

Attribute name and its value to be associated with the specific node in the core schema are defined by **AttributeName** and **AttributeValue** pairs. **AttributeValue** must appear just after **AttributeName**. These pairs are optional.

AttributeName is specified by the following sequence;

Reserved word (**attributeName**) and immediately followed by a colon(:), a space or a tab, a quotation mark("), text string name of the attribute, a quotation mark(") and immediately followed by a semicolon(;).

AttributeValue is specified by the following sequence;

Reserved word (**attributeValue**) and immediately followed by a colon(:), a space or a tab, a quotation mark("), text string expresses the attribute value corresponds to the attribute name, a quotation mark(") immediately followed by a semicolon(;).

White space includes one or more of the following: The space character, the carriage return, the line feed, the tab character.

The following list shows the reserved words for this metadata coding.

List.1 Reserved words

Reserved words
formatID
nodeName
attributeName
attributeValue

Example of metadata instance coding for DAVIC core schema is in Annex B titled "Example of Metadata Instance of DAVIC Core Schema".

5.1.6.4 Metadata Filtering

5.1.6.4.1 Requirements

Preference based metadata filtering is the key technology for the consumer set-top based multimedia retrieval systems. For implementing those systems, the following requirements should be considered.

1) Efficient metadata filtering

Set-top unit is often implemented under some strict minimum resource constraints. The processing power and memory resource required for filtering computations are expected to be minimized.

2) Dedicated metadata broadcast delivery protocol and access interface

Only the metadata should be delivered through some dedicated channel or stream. For filtering efficiency, the stream of metadata expected to be isolated from the stream of the multimedia content packages. The delivery protocol may be completely customized to meet the needs of the application and should reflect performance concerns.

3) Compact filtering mask coding (metadata compression scheme)

To reduce the cost of metadata filtering, a pre-filtering mechanism is required at lower level devices. Most of the metadata format currently being standardized is based on the textual notation using some mark-up languages. Parsing the textual notation introduces some processing overhead compared to processing the fixed record type structure so called a binary representation of index. The filter architecture would be designed to maximize search efficiency and it has an instruction set optimized for efficient execution against a high-speed, synchronous, structured input data stream. Associated with the metadata format conversion from textual notation to binary notation, some notification mechanism to announce the semantics of the bit sequence are required to enable every set-top to understand the converted indexes.

4) Generalized metadata package format

Metadata format is now standardized in various organizations, such as W3C, DVB. To implement efficient metadata filters, generalized package format understood by all set-tops should be standardized. This package should be able to encapsulate metadata expressed by any kind of syntax now being developed by other standardization organizations

5.1.6.4.2 Metadata Package Format

The followings are components expected to be included in generalized metadata package format. Major components are filtering mask, original metadata and locator. (Figure1)

- Filtering Mask
- Original Metadata
- Locator

Filtering Mask: Filter oriented metadata enables set-top hardware level pre-filtering. Generally a filter oriented metadata is encoded in the sequence of bit pattern in order to be handled by the hardware filter straightforward. It has structured binary representation. The semantics of bit sequence is specified in mask schema derived from original metadata schema.

Original Metadata: Inserted if required, for more accurate filtering. It is generally a set of attribute-value pairs. RDF encoded in XML is well known example of original metadata description format for Internet resources. It has rich and detail description of the contents or services. This part is not required in case all vocabularies in original metadata is mapped onto filtering mask.

Locator: Address information of the content for which this metadata package describes. The UPI (Uniform Program Identifier) or URL are typical examples of the locator.

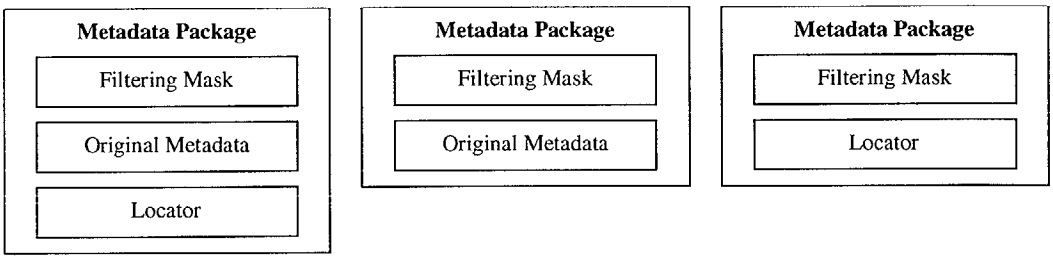


Figure 32 : Metadata Package

Figure 32. shows three types of combinations of the metadata package components. The first one is for filtering the original metadata and locator. The second one is for filtering only the original metadata which may or may not include the locating information. The third one is for filtering only the locator. For the third case, whole information in the original metadata is mapped into the filtering mask part, or filtering mask part has no metadata information but some compressed form of the locator, which is used only for the efficient extraction of the locator part from broadcast channel. The structure of those components are adopted to enable layered filtering described in Figure 33.

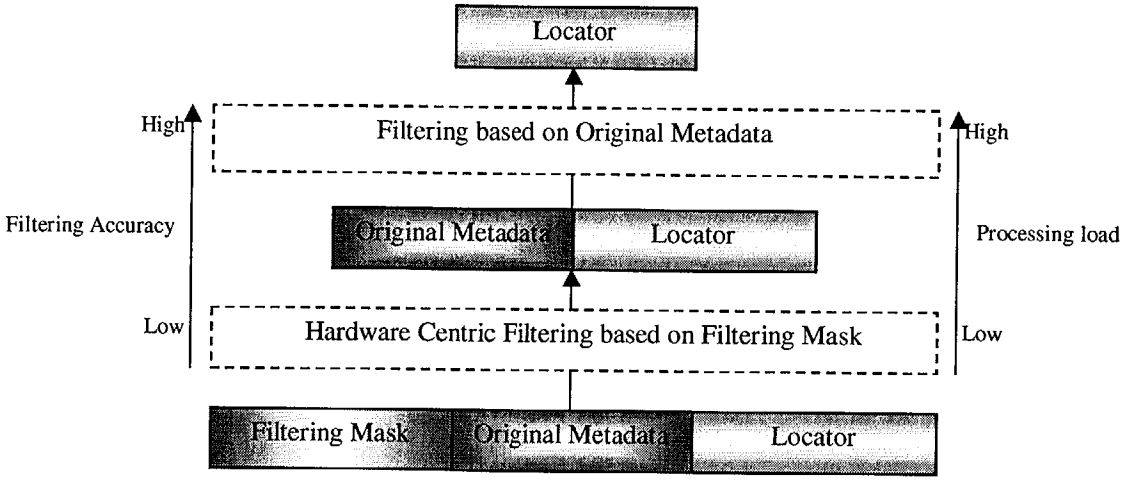


Figure 33 : Layered Filtering for Preferred Locator Extraction

5.1.6.4.3 Filtering Mask Conversion

Filtering mask would be recognized as the compressed form of the original metadata description. It is desired that the all of the attribute-value pairs of original metadata should be converted onto the bit pattern of the filtering mask. See Figure 34. But in some cases, this conversion process is hard to design due to the complexity of metadata description framework, syntax and vocabularies. A part of the original metadata, corresponds to a part of the schema which is easy to be converted, should be used as a source of the filtering mask in those cases. See Figure 35.

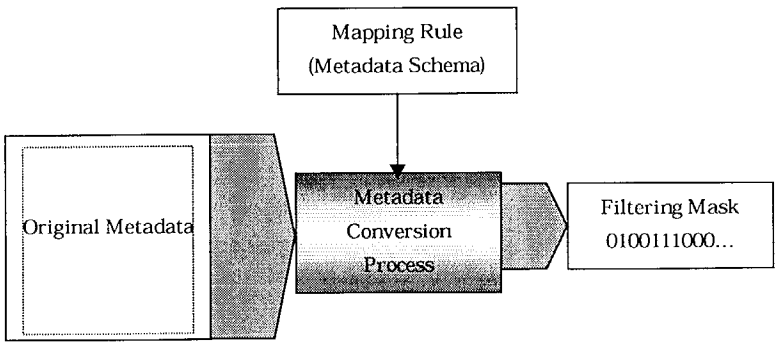


Figure 34 : All of Original Metadata is mapped onto bit pattern

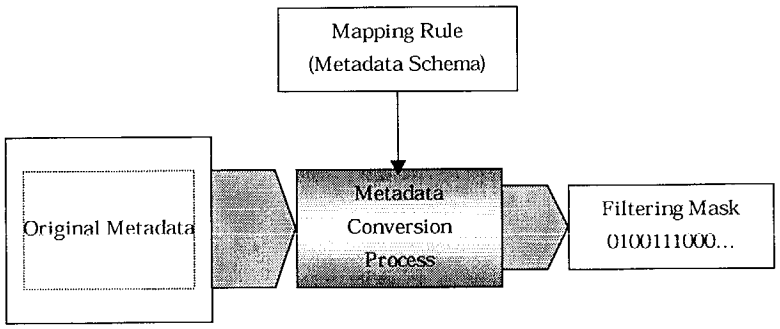


Figure 35 : A Part of Original Metadata is mapped onto bit pattern

5.1.6.4.4 Filtering Mask Conversion Example

This section explains how the metadata expressed in RDF data model is converted onto filter centric bit sequence structure. RDF is simple but expressive data model for metadata.

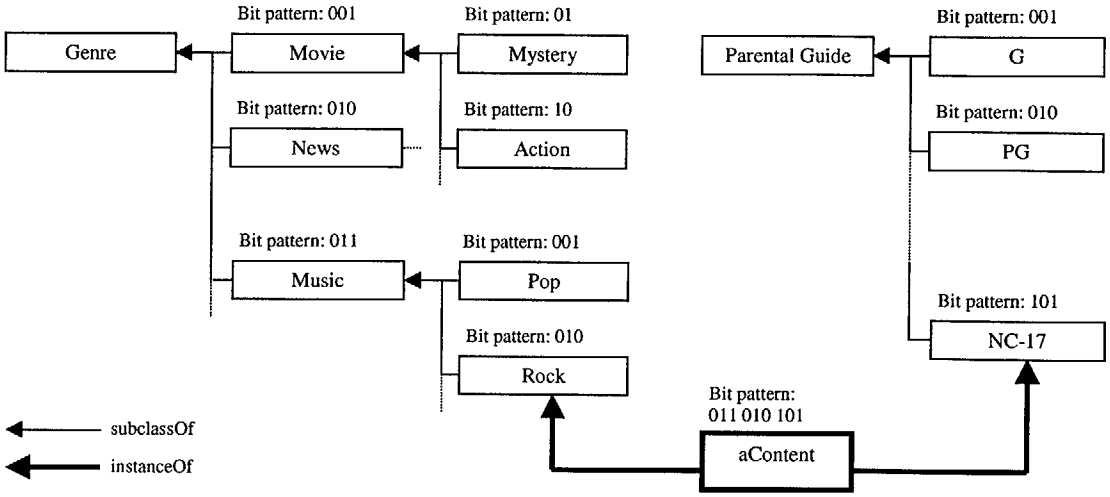


Figure 36 : Class hierarchies defined in RDF

Figure 36 shows simple class hierarchies defined in RDF data model. There are categories such as "Parental Guide", whose allowable values are G, PG, NC-17, and "Genre", whose values are movies, mystery, action and so on. Any content with which metadata attached is an instance of some classes in this hierarchy.

This class hierarchy schema would be converted onto a mask schema expressed in machine understandable format by assigning offset and bit patterns to each node in this structure. Figure 37 shows an example of mask schema corresponding to this hierarchy described in Figure 37. In this schema, format ID identifies the encoding format of encapsulated metadata. Schema ID is a unique identifier for the conversion algorithm from original metadata schema to filtering mask bit sequence. Uniqueness of format ID and schema ID should be managed by the service provider.

Figure 36 shows that bit pattern "011" is assigned to the node named "music", bit pattern "010" is assigned to the node named "rock" and "NC-17" has an assigned bit pattern "101". A node named "aContent" represents exact content whose genre is rock and parental guide is NC-17. Namely "aContent" is an instance of class "rock" and "NC-17". So the generated filtering mask bit pattern for "aContent" is "011 010 101", the first 6 bits pattern "011 010" means that its genre and the last 3 bits "101" means its parental guide. Thus, "aContent" has a genre of "music/rock". Also it is classified in "NC-17" of parental guide categorization.

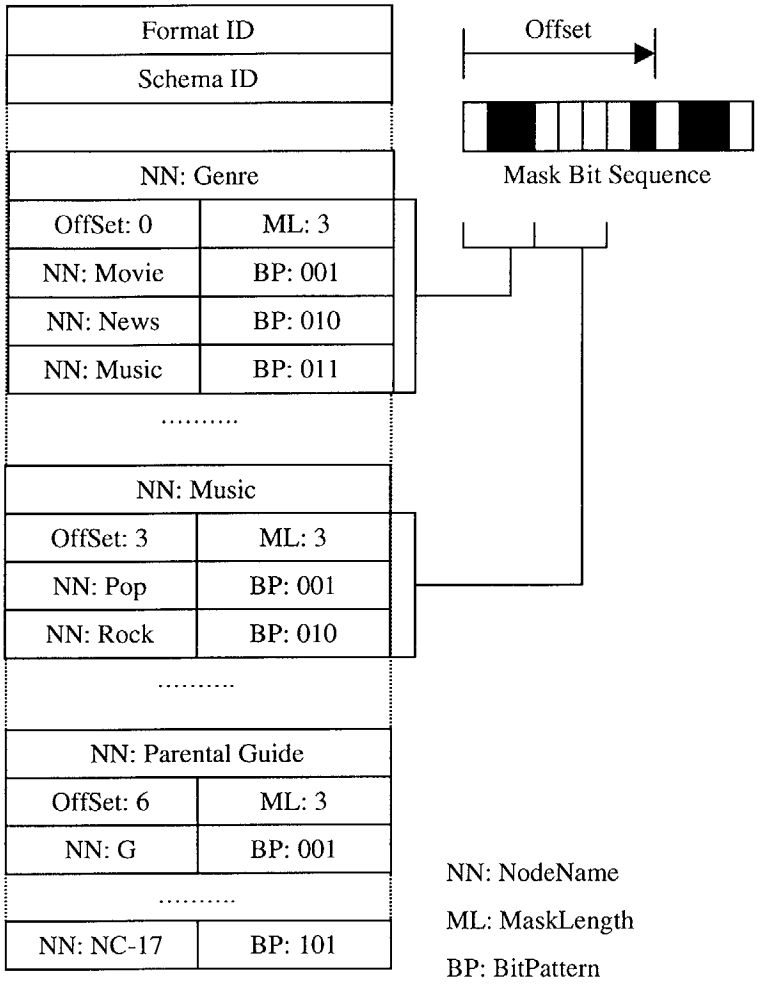


Figure 37 : Mask Schema

Figure 38 shows the filtering mask structure of metadata package encapsulating the original metadata corresponds to "aContent". Definition of format ID and schema ID are the same as specified in the description of mask schema in Figure6. Mask bit sequence is interpreted based on the mask schema uniquely identified by those IDs.

Format ID
Schema ID
Mask Bit Sequence (011010101)

Figure 38 : Filtering Mask Structure

Prior to the transmission of the metadata package including this filtering mask, the corresponding metadata schema should be sent to or should be resident on the set-top. The set-top can realize the change of the metadata

schema by fetching the FormatID and SchemaID of this structure and comparing them with those assigned to the mask schema already stored or resident on the set-top.

5.1.6.4.5 Coding Syntax for Metadata Schema

The mask schema (called metadata schema in this specification) specifies the mapping rule between the original metadata schema and the filtering mask bit sequence. See Figure 39. This section specifies a simple syntax for the metadata schema notation. This syntax is for specifying mapping rules between hierarchical structured nodes, hierarchical structured portion of the original metadata schema of various kinds of format, and their assigned bit patterns and their address within the mask bit sequence.

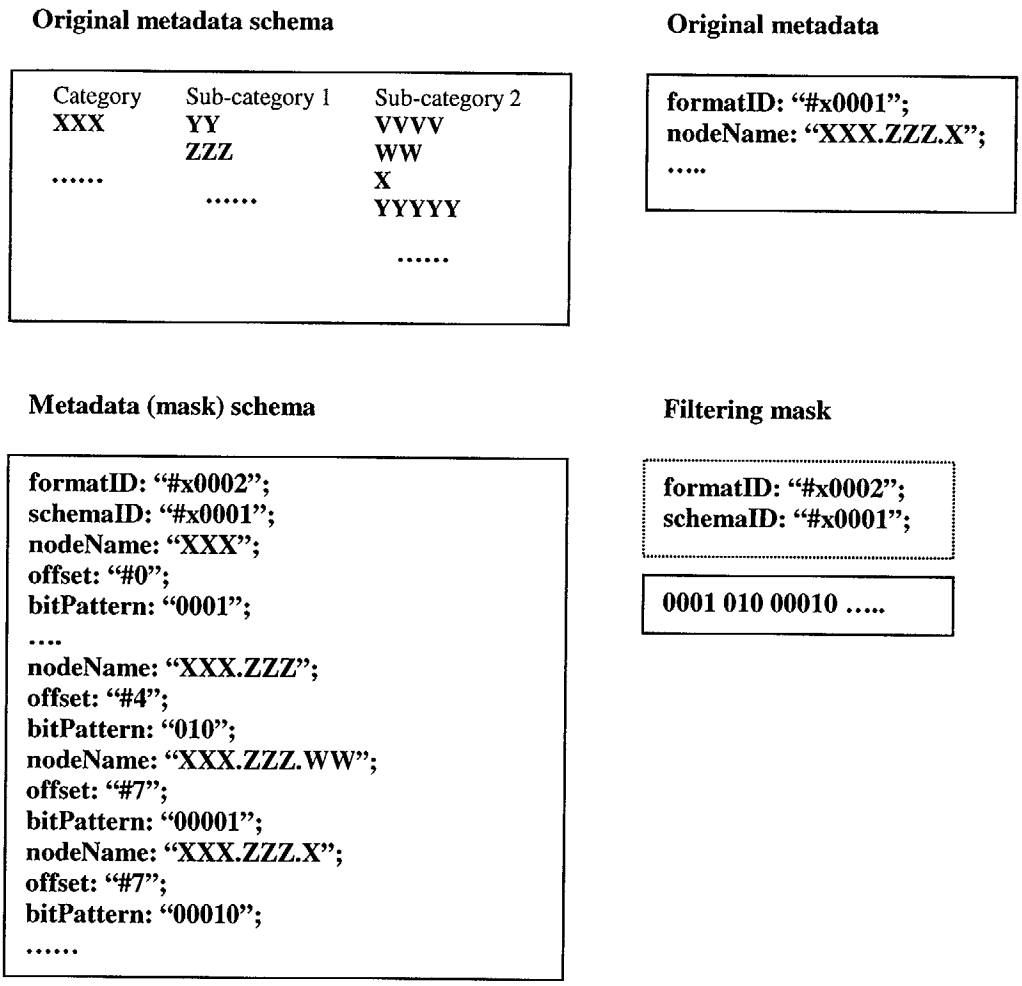


Figure 39 : Metadata schema and its instance

5.1.6.4.5.1 Syntax Description

Character set:

Characters are based on the standard ISO10646 character encoding scheme or Unicode. The character set used for metadata broadcasting system is predetermined by the service provider.

Comments are expressed by the text within the /* .... */ code.

The schema definition has the following sequence;

FormatID, white space,

SchemaID, white space,

Set of (

NodeName, white space,

Offset, white space,

BitPattern, white space

)

FormatID and **SchemaID** is an unique identifier of this schema. The value of those IDs are managed by the service provider. **FormatID** identifies this schema syntax. **SchemaID** identifies the original metadata schema from which this mapping rule is derived.

FormatID is specified by the following sequence;

Reserved word (**formatID**) and immediately followed by a colon(:), a space or a tab, a quotation mark("), hexadecimal integer preceded by a pond sign and the literal string x (#x), a quotation mark(") and immediately followed by a semicolon(;).

SchemaID is specified by the following sequence;

Reserved word (**schemaID**) and immediately followed by a colon(:), a space or a tab, a quotation mark("), hexadecimal integer preceded by a pond sign and the literal string x (#x), a quotation mark(") and immediately followed by a semicolon(;).

NodeName expresses the node name of the hierarchical structured nodes corresponds to the metadata descriptor. **Offset** indicates the offset from the top bit of the mask bit sequence. **BitPattern** expresses the bit pattern assigned to this node.

NodeName is specified by the following sequence;

Reserved word (**nodeName**) and immediately followed by a colon(:), a space or a tab, a quotation mark("), text string name of the node in the unix file system naming syntax, a quotation mark(") and immediately followed by a semicolon(;).

Offset is specified by the following sequence;

Reserved word (**offset**) and immediately followed by a colon(:), a space or a tab, a quotation mark("), decimal integer preceded by a pond sign(#), a quotation mark(") and immediately followed by a semicolon(;).

BitPatten is specified by the following sequence;

Reserved word (**bitPattern**) and immediately followed by a colon(:), a space or a tab, a quotation mark("), text string expresses assigned bit pattern, a quotation mark(") immediately followed by a semicolon(;).

Bit pattern is specified by bit string, left bit first (bslbf) style.

White space include one or more of the following: The space character, the carriage return, the line feed, the tab character.

The following list shows the reserved words for this schema expression.

List.1 Reserved words

Reserved words
formatID
schemaID
nodeName
offset
bitPattern

Example metadata schema coding for mapping between original metadata schema of DAVIC core schema and the mask bit sequence is in Annex C titled "Example of Metadata Schema for DAVIC Core Schema".

5.1.6.4.6 Coding Syntax for Metadata Schema Extension

Extensions to the metadata schema can be made by sending a bit pattern and the associated node in the extended schema to the client. See Figure 40. Note that it is not necessary to send the complete schema when relying on the base metadata schema is already in the client. Only the schema describes the additional portion of the base metadata schema shall be sent to the client.

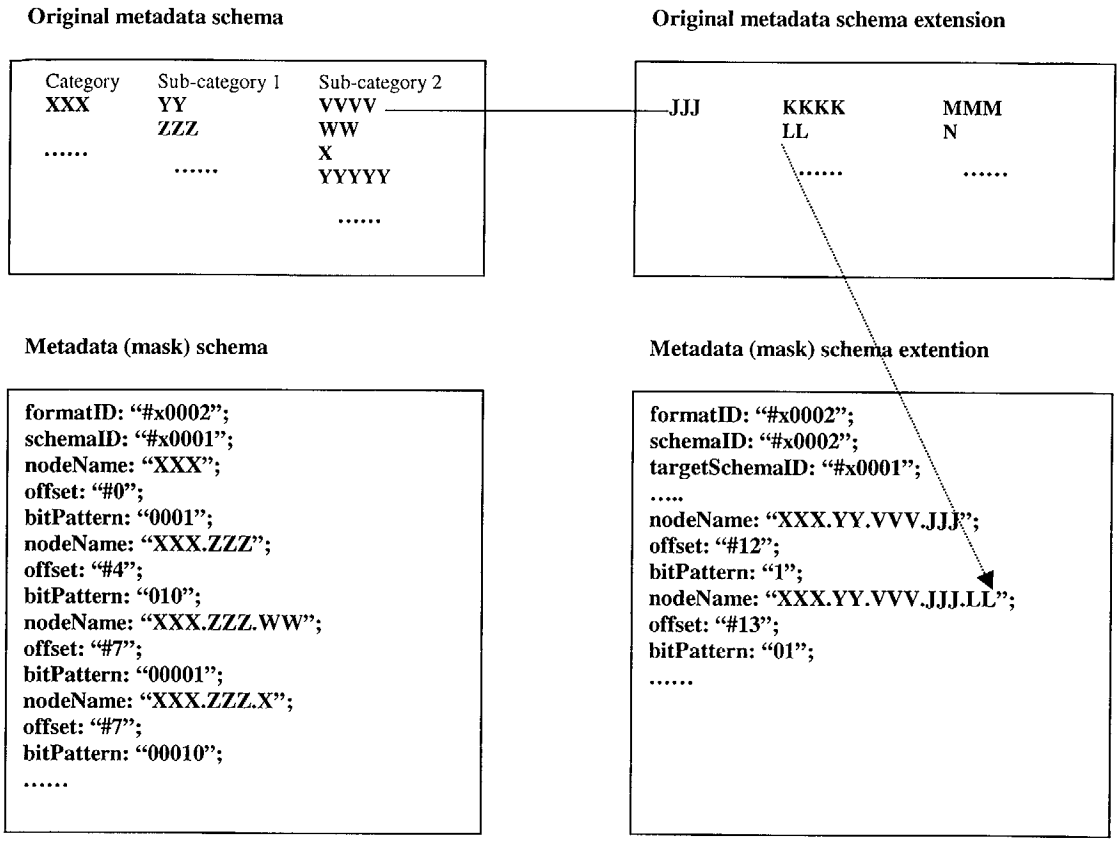


Figure 40 : Metadata Schema Extension

The syntax is the same as the description in the previous section except for the additional means for extending the nodes hierarchy.

5.1.6.4.6.1 Syntax Description

The schema has the following sequence;

FormatID, white space,

SchemaID, white space,

TargetSchemaID, white space,

Set of (

NodeName, white space,

Offset, white space,

BitPattern, white space

)

Definition of **FormatID**, **SchemaID**, **NodeName**, **Offset**, and **BitPattern** are the same as the definition in the previous section.

TartgetSchemaID is specified by the following sequence;

Reserved word (**tergetSchemaID**) and immediately followed by a colon(:), a space or a tab, **SchemaID** and immediately followed by a semicolon(;).

SchemaID in this **TergetSchemaID** part identifies that the target schema to which this extension is applied . After those extension is applied to the schema, it will be assigned a new **SchemaID** specified at the first **SchemaID** part of this file. Extension of nodes and their assigned bitpattern are specified in the following set of (**Node_Name**, **Offset** and **BitPattern**).

Reserved word "targetSchemaID" is added to the list.1 defined in the previous section.

List.2 Reserved words

Reserved words
formatID
schemaID
targetSchemaID
nodeName
offset
bitPattern

Coding example of this application specific schema extension is in Annex D titled "Example of Metadata Schema Extension for DAVIC Core Schema".

5.1.6.5 Metadata carriage in MPEG2 TS

5.1.6.5.1 Metadata Package Format

The metadata package is encapsulated in private section and delivered on the MPEG2 system stream on the DVB complaint delivery platform for satellite broadcast network. See Figure 41. The filtering mask bit sequence should be mapped within the first 10 bytes of the section header. It insures efficient metadata package extraction at DEMUX level filtering. Because of the current constraints of DEMUX architecture already on the market, only a few bytes out of 10 bytes could be used for this purpose. But in the future, this constraint is expected disappear by introducing the advanced DEMUX hardware devices.

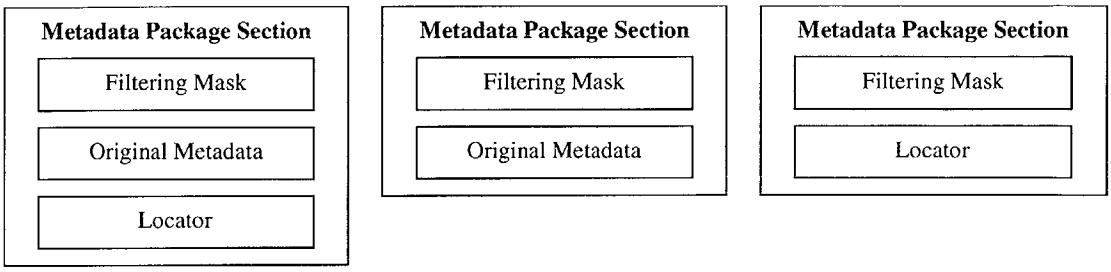


Figure 41 : Metadata Package Section

5.1.6.5.1.1 Metadata Package Section

The metadata package section has the following format.

Table 5-18. Syntax of Metadata Package Section

Syntax	**Nr of**	**Identifier**
metadata_package_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_for_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf

filtering_mask _bytes	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
for (j=2;j<filtering_mask_length -2;j++){		
filtering_mask_bytes	8	uimsbf
}		
descriptors_loop_length	12	uimsbf
reserved	4	bslbf
for (j=0;j<descriptors_loop_length;j++){		
descriptor()		
}		
CRC_32	32	rpchof
}		

Semantics for the metadata package section:

table_id: The table_id is an 8-bit field which identifies each descriptor.

section_syntax_indicator: The section_syntax_indicator is a 1-bit field which shall be set to "1".

section_length: This is a 12-bit field, the first two bits of which shall be "00". It specifies the number of bytes of the section, starting immediately following the section_length field and including the CRC. The section_length shall not exceed 1021 so that the entire section has a maximum length of 1024 bytes.

filtering_mask_bytes: This is a 16-bit field which stores the first two bytes of Filering Mask. The rest of the filtering mask bytes are conveyed in the fields immediately after last_section_number field. The total length of the filtering mask field is specified in filtering_mask_descriptor placed in component loop of PMT. The FormatID and SchemaID are also conveyed in component loop of PMT. This means that all the filtering mask of metadata package section in the same elementary stream have the same FormatID and SchemaID.

version_number: This 5-bit field is the version number of the sub_table. The version_number shall be incremented by 1 when a change in the information carried within the sub_table occurs. When it reaches value "31", it wraps around to "0". When the current_next_indicator is set to "1", then the version_number shall be that of the currently applicable sub_table. When the current_next_indicator is set to "0", then the version_number shall be that of the next applicable sub_table.

current_next_indicator: This 1-bit indicator, when set to "1" indicates that the sub_table is the currently applicable sub_table. When the bit is set to "0", it indicates that the sub_table sent is not yet applicable and shall be the next sub_table to be valid.

section_number: This 8-bit field gives the number of the section. The section_number of the first section in the sub_table shall be "0x00". The section_number shall be incremented by 1 with each additional section with the same table_id, filtering_mask_bytes, and original_network_id.

last_section_number: This 8-bit field specifies the number of the last section (that is, the section with the highest section_number) of the sub_table of which this section is part.

descriptors_loop_length: This 12-bit field gives the total length in bytes of the following descriptors. In the descriptor of this metadata package section, both any number of metadata_descriptor() and locator_descriptor() are conveyed.

descriptor(): Any number of metadata_descriptor() or Locator_descriptor().

CRC_32: This is a 32-bit field that contains the CRC value that gives a zero output of the registers in the decoder defined in annex B of DVB-SI specification after processing the entire section.

5.1.6.5.1.2 Metadata Descriptor

The metadata descriptor encapsulates the original metadata. If this metadata package is used for carrying the metadata instance of DAVIC core schema and its extension, the instance in the form of textual notation is carried in this descriptor and filtering mask bit sequence based on metadata schema corresponds to DAVIC core schema is conveyed in the filtering mask part of the same section. The metadata in this descriptor can be omitted if all the information in metadata instance is mapped onto the filtering mask part.

Table 5-19. Syntax of Metadata Descriptor

Syntax	No. of Bits	Identifier
metadata_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
for(i=0;i<N;i++){		
char	8	uimsbf
}		
}		

Semantics for the metadata descriptor:

descriptor_tag: The descriptor tag is an 8-bit field which identifies each descriptor.

descriptor_length: The descriptor length is an 8-bit field specifying the total number of bytes of the data portion of the descriptor following the byte defining the value of this field.

char: This is an 8-bit field, sequence of which conveys the original metadata instance bytes.

5.1.6.5.1.3 Locator Descriptor

The locator descriptor encapsulates the locator used for address resolution from locator logical name to real address for the contents or service entity.

Table 5-20. Syntax of Locator Descriptor

Syntax	No. of Bits	Identifier
Locator_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
locator_syntax_identifier	8	uimsbf
for(i=0;i<N;i++){		
char	8	uimsbf
}		
}		

Semantics for the locator descriptor:

descriptor_tag: The descriptor tag is an 8-bit field which identifies each descriptor. TBD.

descriptor_length: The descriptor length is an 8-bit field specifying the total number of bytes of the data portion of the descriptor following the byte defining the value of this field.

locator_syntax_identifier: This is an 8-bit field which identifies the syntax of the content locator. The value is defined in the following table.

Table 5-21. Definition of locator_syntax_identifier

Value	Format description
TBD	UPI

TBD	URL

char: This is an 8-bit field, a sequence of which conveys the content locator bytes.

5.1.6.5.1.4 Filtering Mask Descriptor

This descriptor shall be placed in component loop of PMT not in Metadata package section. This descriptor specifies the corresponding elementary stream conveys the metadata package section and its metadata schema. The descriptor has the information on how many bytes of the header part of the section are used for the filtering purpose. The type of this elementary stream which carries matadata packages is specified by data_broadcast_id_descriptor defined in DVB-SI. The value of Data_broadcast_id is specified by the service provider.

Table 5-22. Syntax of Filtering Mask Descriptor

Syntax	**No. of Bits**	**Identifier**
filtering_mask_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
filtering_mask_byte_length	8	uimsbf
formatID	16	uimsbf
schemaID	16	uimsbf
}		

Semantics for the metadata descriptor:

descriptor_tag: The descriptor tag is an 8-bit field which identifies each descriptor.

descriptor_length: The descriptor length is an 8-bit field specifying the total number of bytes of the data portion of the descriptor following the byte defining the value of this field.

filtering_mask_byte_length: The value of this field indicates how many bytes shall be used for the filtering.

formatID, schemaID: The combination of those values specifies the format and conversion process between the original metadata and filtering mask bytes. Uniqueness of those values are managed by the service provider.

5.1.6.5.1.5 Metadata Schema (Mask Schema) Section

Metadata schema or Metadata Schema Extension are carried on the metadata schema section.

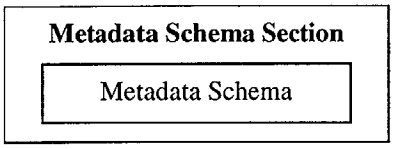


Figure 42 : Metadata Schema Section

The metadata schema section has the following format.

Table 5-23. Syntax of Metadata Schema (Mask Schema) Section

Syntax	Nr of bits	Identifier
metadata_schema_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_for_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
reserved	16	uimsbf
reserved	2	uimsbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
descriptors_loop_length	12	uimsbf
reserved	4	bslbf
for (j=0;j<descriptors_loop_length;j++){		
descriptor()		
}		
CRC_32	32	rpchof
}		

Semantics for the metadata schema section:

table_id: The table_id is an 8-bit field which identifies each descriptor.

section_syntax_indicator: The section_syntax_indicator is a 1-bit field which shall be set to "1".

section_length: This is a 12-bit field, the first two bits of which shall be "00". It specifies the number of bytes of the section, starting immediately following the section_length field and including the CRC. The section_length shall not exceed 1021 so that the entire section has a maximum length of 1024 bytes.

version_number: This 5-bit field is the version number of the sub_table. The version_number shall be incremented by 1 when a change in the information carried within the sub_table occurs. When it reaches value "31", it wraps around to "0". When the current_next_indicator is set to "1", then the version_number shall be that of the currently applicable sub_table. When the current_next_indicator is set to "0", then the version_number shall be that of the next applicable sub_table.

current_next_indicator: This 1-bit indicator, when set to "1", indicates that the sub_table is the currently applicable sub_table. When the bit is set to "0", it indicates that the sub_table sent is not yet applicable and shall be the next sub_table to be valid.

section_number: This 8-bit field gives the number of the section. The section_number of the first section in the sub_table shall be "0x00". The section_number shall be incremented by 1 with each additional section with the same

last_section_number: This 8-bit field specifies the number of the last section (that is, the section with the highest section_number) of the sub_table of which this section is part.

descriptors_loop_length: This 12-bit field gives the total length in bytes of the following descriptors. In the descriptor of this metadata schema section, any number of metadata_schema_descriptor() are conveyed.

descriptor(): Any number of metadata_schema_descriptor().

CRC_32: This is a 32-bit field that contains the CRC value that gives a zero output of the registers in the decoder defined in annex B of DVB-SI specification after processing the entire section.

5.1.6.5.1.6 Metadata Schema Descriptor

The metadata schema descriptor conveys metadata schema or metadata schema extension which specifies the mapping between the original metadata and filtering mask bytes. Any number of metadata schemas could be in a metadata_schema_section.

Table 5-24. Syntax of Metadata Schema Descriptor

Syntax	No. of Bits	Identifier
metadata_schema_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
formatID	16	uimsbf
schemaID	16	uimsbf
for(i=0;i<N;i++){		
char	8	uimsbf
}		
}		

Semantics for the metadata schema descriptor:

descriptor_tag: The descriptor tag is an 8-bit field which identifies each descriptor.

descriptor_length: The descriptor length is an 8-bit field specifying the total number of bytes of the data portion of the descriptor following the byte defining the value of this field.

formatID, schemaID: The combination of those values specifies the data format and conversion process between the original metadata and filtering mask bytes. Uniqueness of those values are managed by the service provider. FormatID specifies the encoding syntax of encapsulated metadata schema. SchemaID is an unique ID for this metadata schema.

char: This is an 8-bit field, a sequence of which conveys the metadata schema bytes. The syntax and the definition of the detail specification of the mapping between the original metadata and filtering mask bytes is specified by the service provider.

Figure 43 shows the tables used for the metadata filtering system.

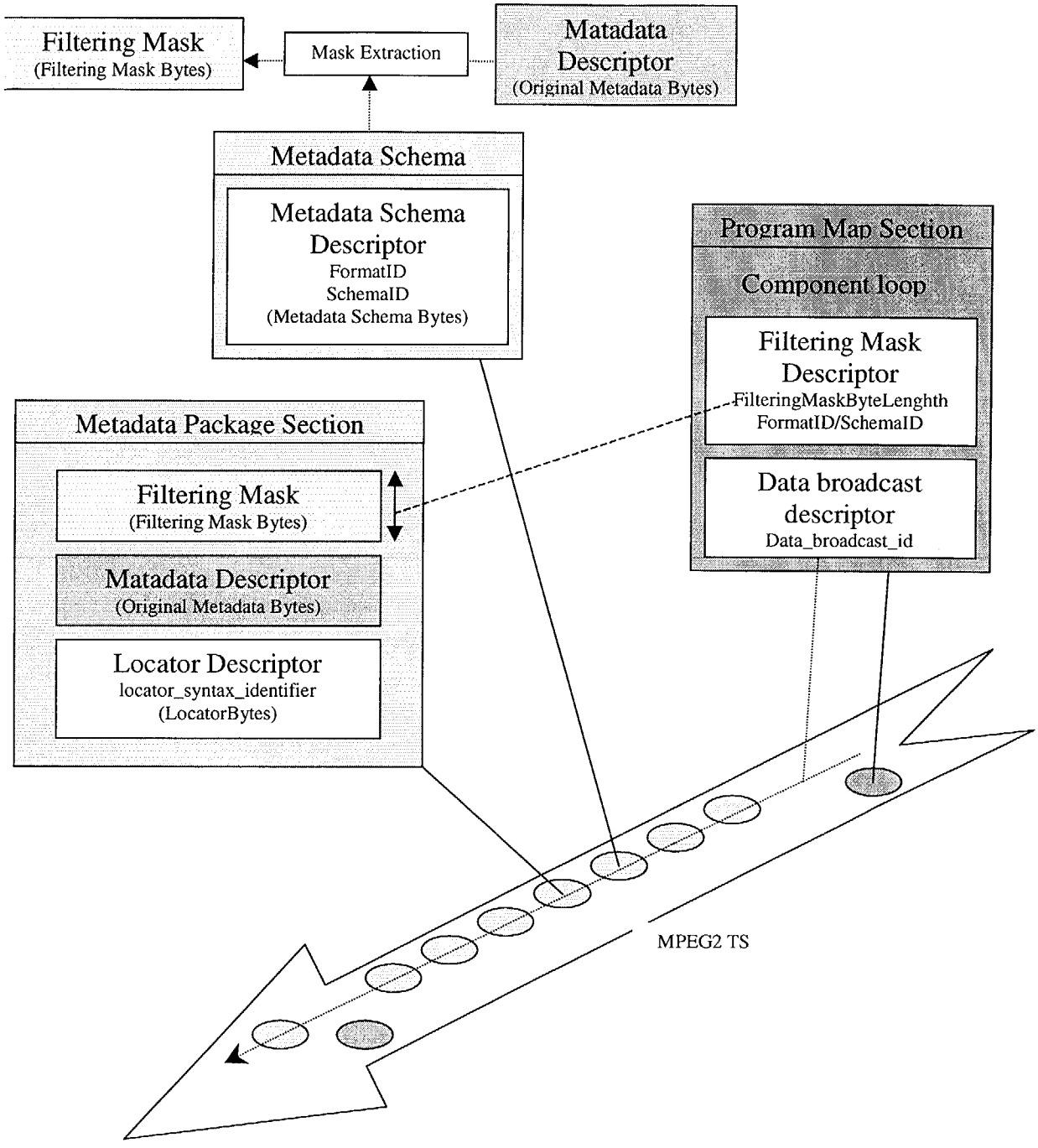


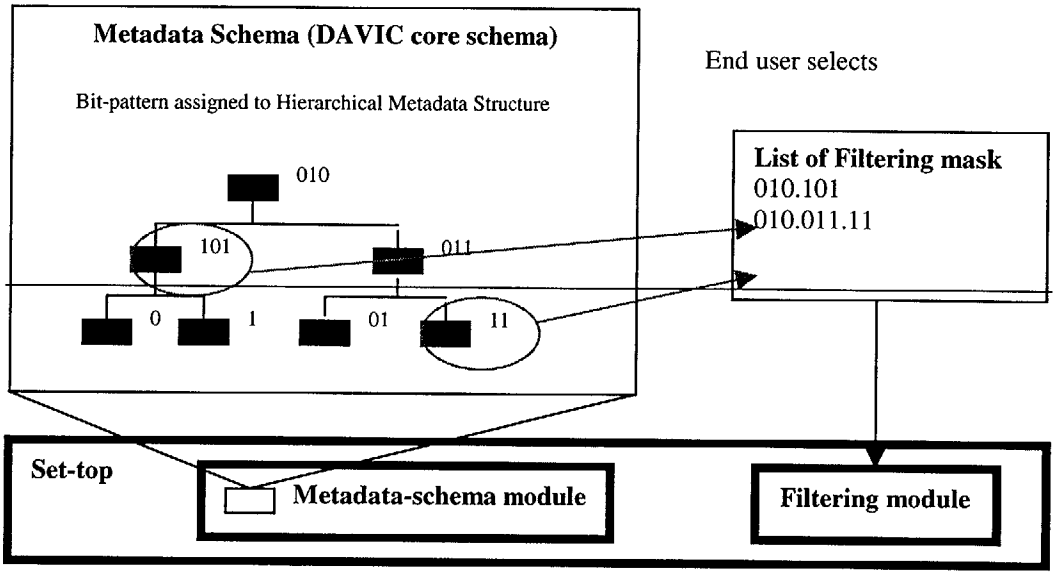
Figure 43 : Tables and descriptors for metadata filtering system

5.1.6.6 Service Scenario Examples

5.1.6.6.1 Filtering mask selection process

1. Set-top on Service Consumer (SC) and Server on Service Provider (SP) share the filtering mask bit-pattern for DAVIC core mask schema. (e.g. 1) Stores the metadata-schema in database of Server on SP and puts metadata-schema on the ROM of Set-top. 2) Set-top on SC retrieves metadata-schema through WWW from the database of Server on SP.)

2. The end-user on Service Consumer selects filtering mask bit-pattern and tells the list of filtering masks to filtering module on his set-top.



5.1.6.6.2 EPG scenario based on DAVIC core schema

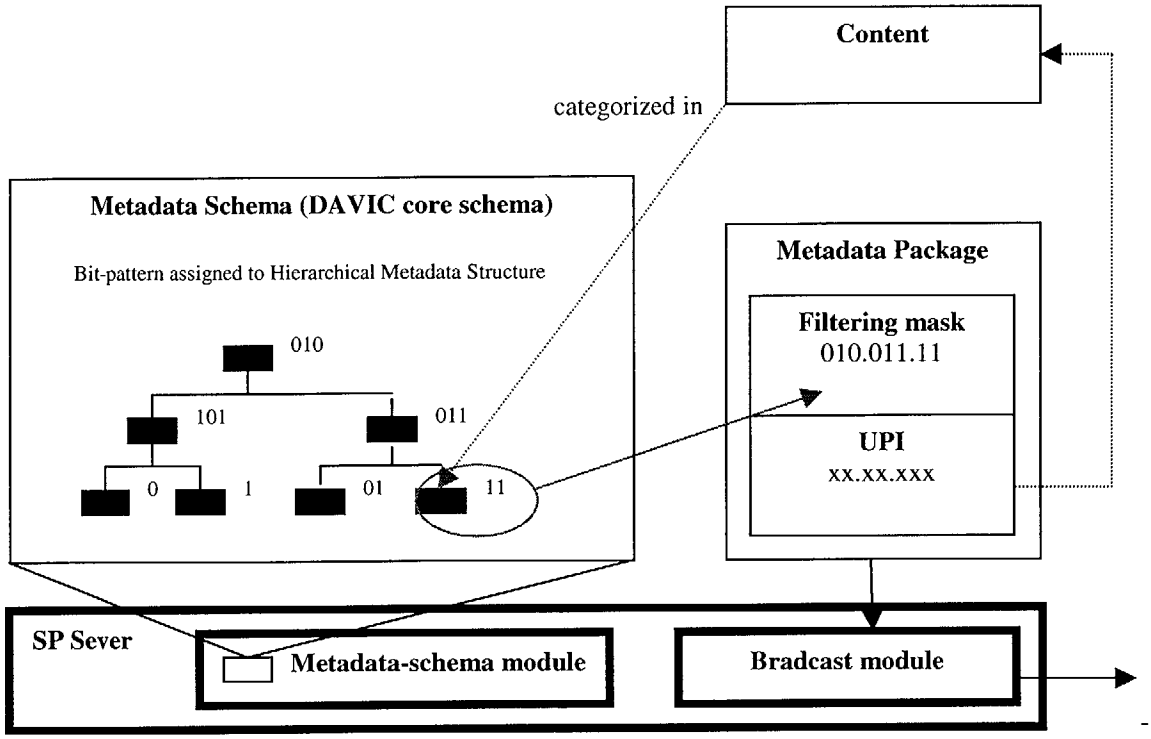
1.Service Provider (SP) creates the content.

2.SP creates the metadata-package.

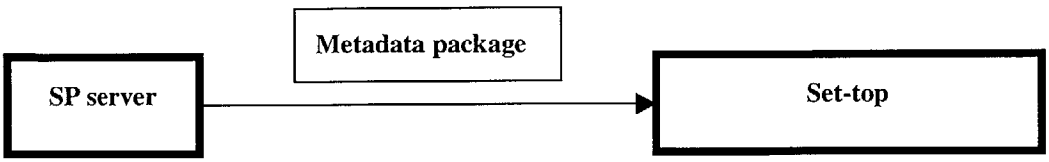
2.1.Selects the metadata category based on the DAVIC core schema.

2.2.Encodes the filtering mask based on the bit-pattern syntax for DAVIC core schema.

2.3.Encodes the UPI for the content.

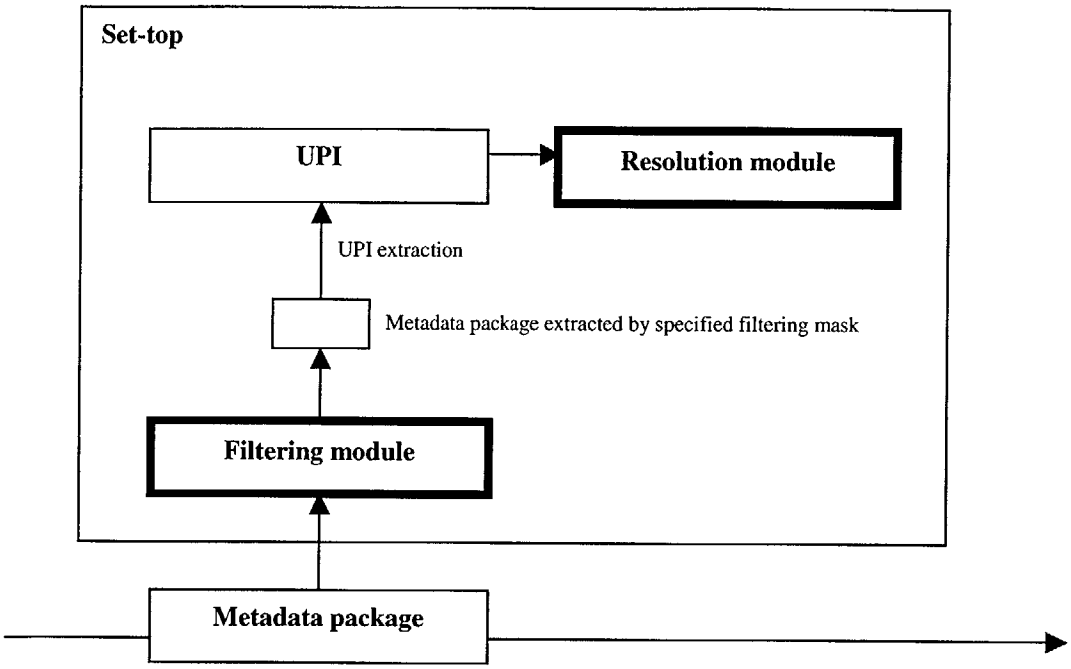


3.SP broadcasts the matadata-package to the set-tops.



4.Set-top on Service Consumer (SC) extracts the matadata-package of his concern based on the filtering mask already set on filtering module on his set-top.

5.Set-top extracts the UPI and asks the UPI resolution module to resolve it.



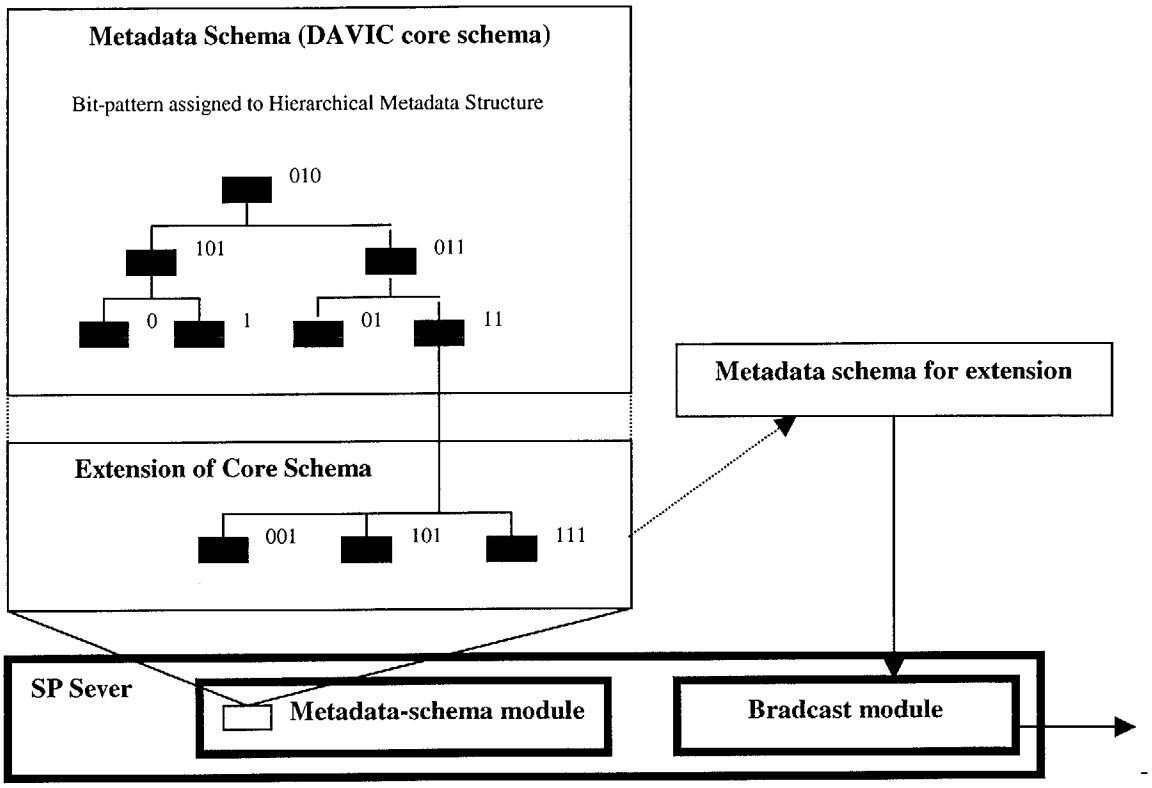
-

6.Resolution module gets the locator(URL) for the content and Set-top captures it by the locator.

5.1.6.6.3 EPG scenario based on extended DAVIC core mask schema

1.SP extends the core schema.

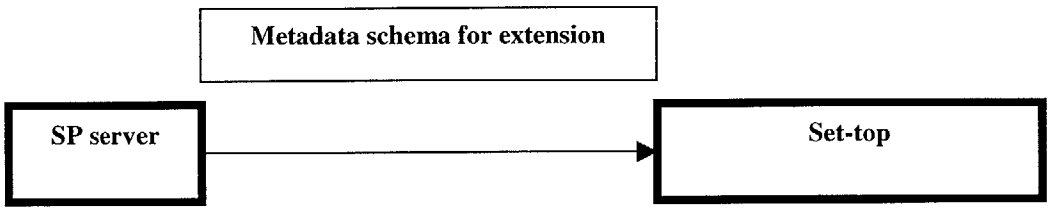
2.SP creates the metadata schema for extended part of core schema called metadata schema for extension.



-

3.SP broadcasts the metadata schema for extension.

-

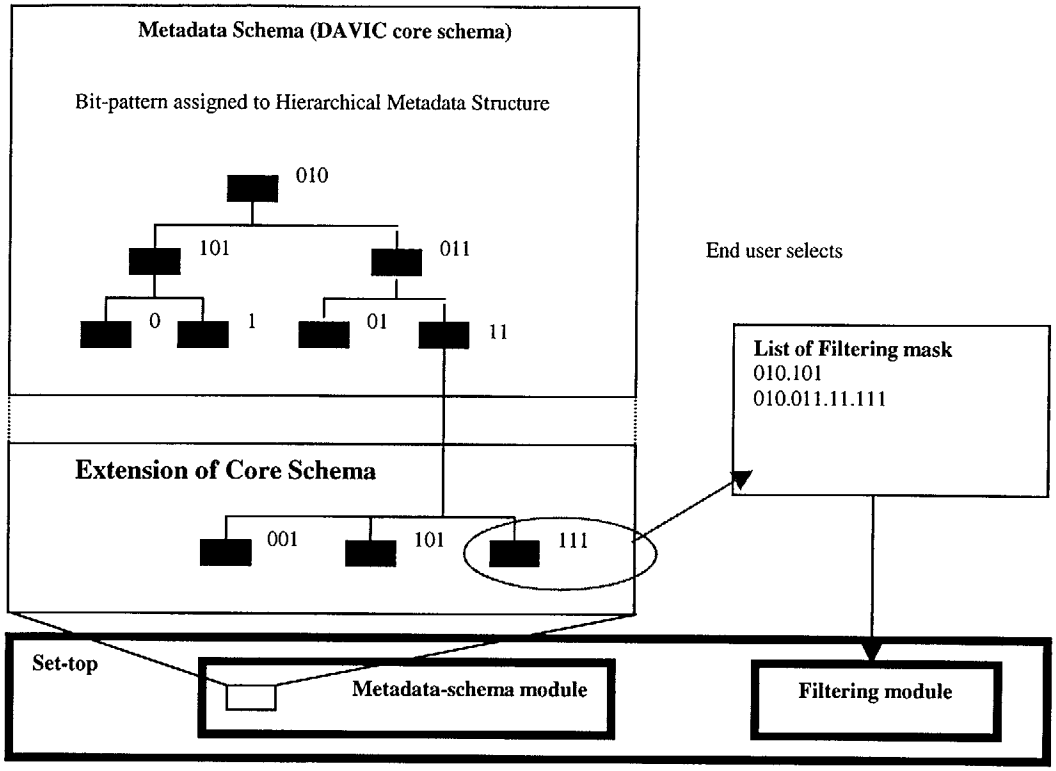


-

4.Set-top on SC extends the core schema resident on schema module by the metadata schema for extension captured through broadcast channel.

5.The end-user on Service Consumer selects filtering mask bit-pattern and tells the list of filtering masks to filtering module on his set-top.

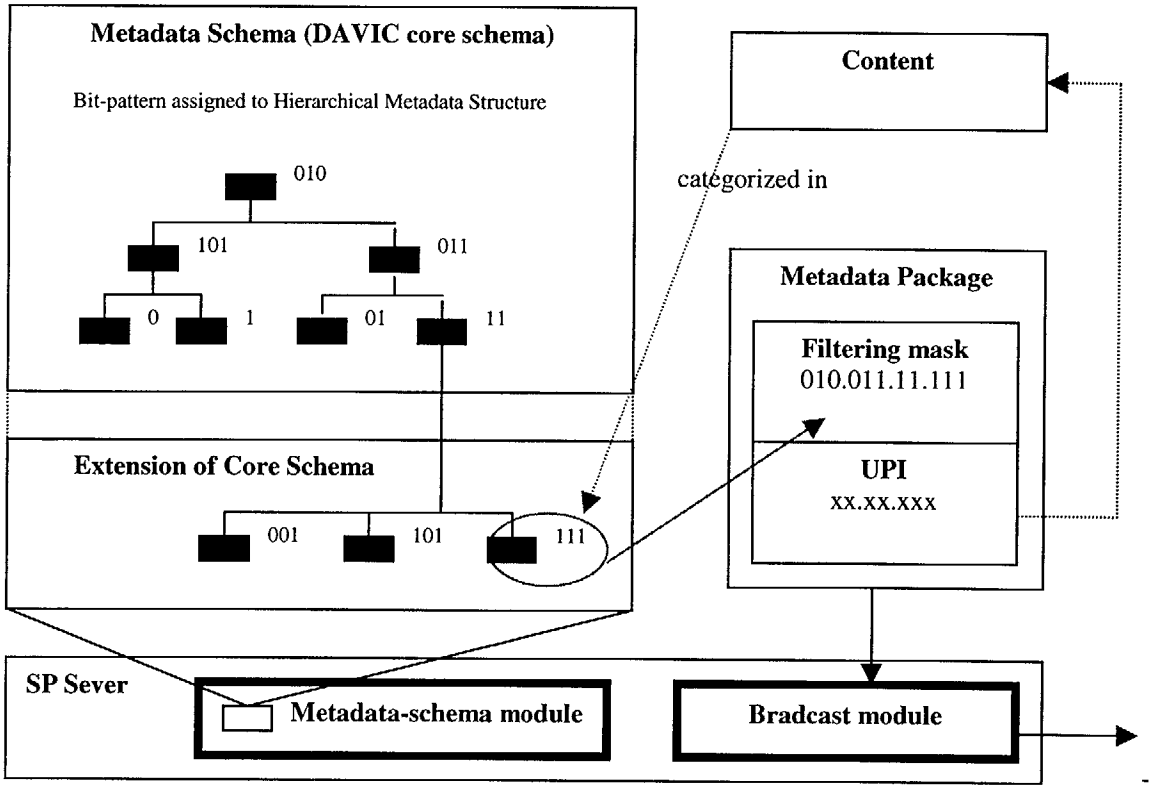
-



-

6.Service Provider (SP) creates the content.

7.SP creates the metadata-package.



-

-

8.SP broadcasts the matadata-package to the set-tops.

9.Set-top on Service Consumer (SC) extracts the matadata-package of his concern based on the filtering mask already set on filtering module on his set-top.

10.Set-top extracts the UPI and asks the UPI resolution module to resolve it.

5.1.6.6.4 EPG scenario based on complex textual metadata

1.Service Provider (SP) creates the content.

2.SP creates complex textual metadata based on DAVIC core schema or other syntax and schema (e.g. RDF, XMLschema, XML). (Assuming that the other metadata schema follows the hierarchical metadata structures defined in DAVIC core schema)

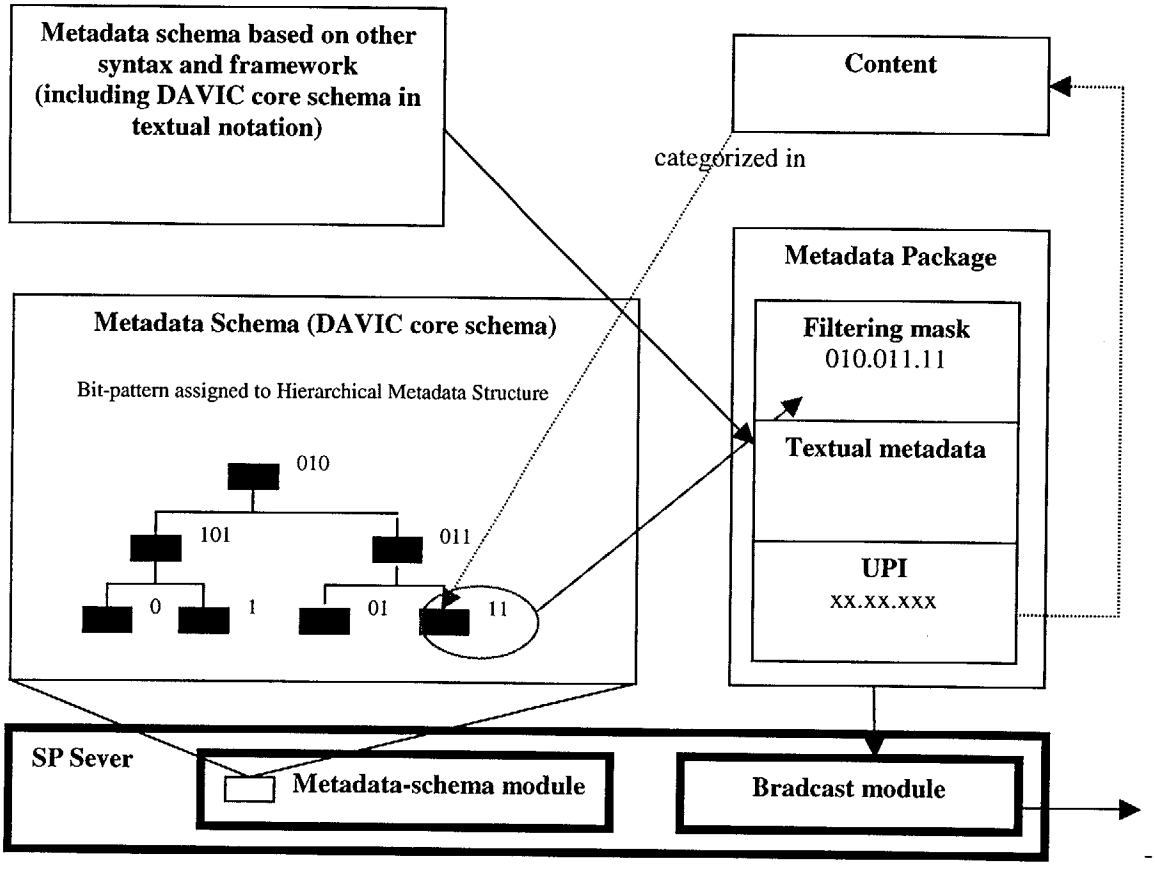
3.SP creates the metadata-package.

3.1.Selects the metadata category based on the DAVIC core schema.

3.2.Encodes the filtering mask based on the bit-pattern syntax for DAVIC core schema.

3.3.Encodes the UPI for the content.

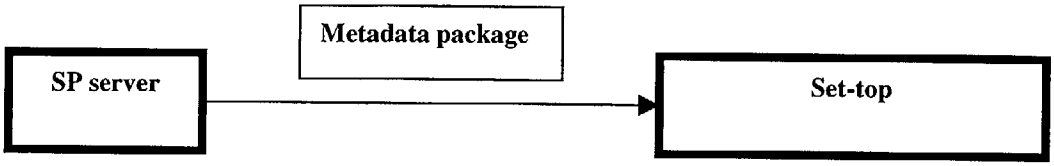
3.4.Includes the complex textual metadata.



-

4.SP broadcasts the matadata-package to the set-tops.

-



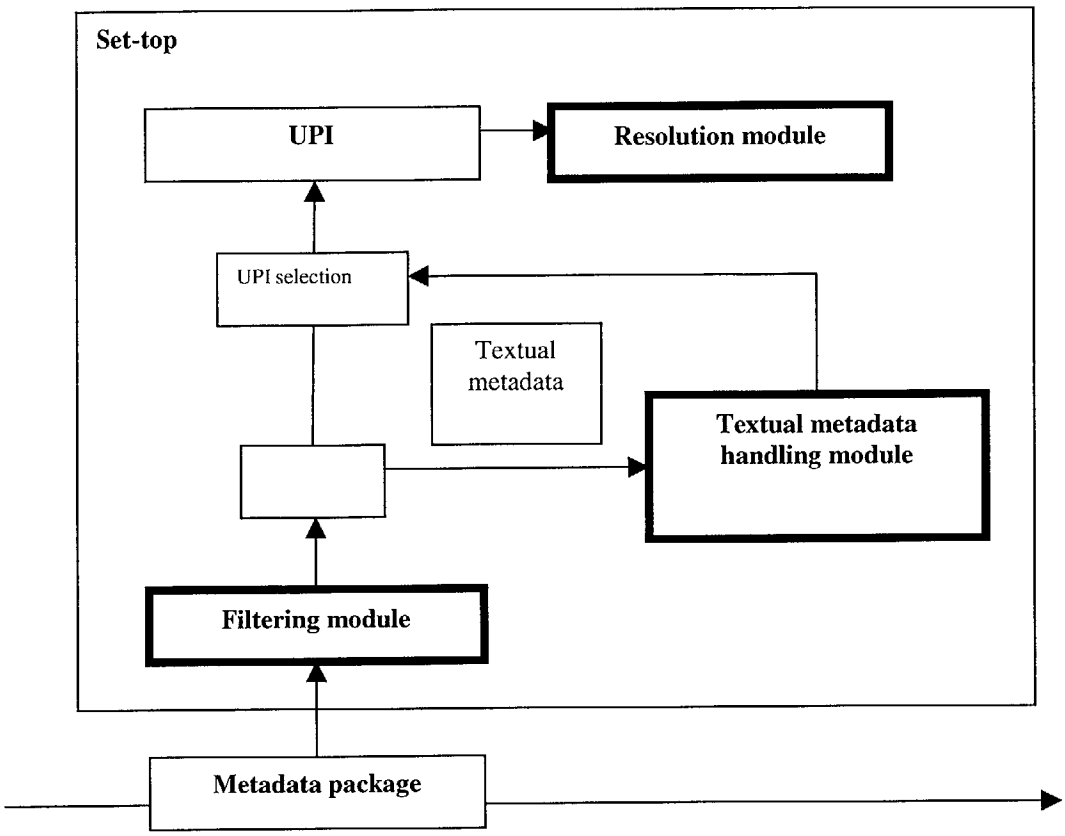
-

5.Set-top extracts the matadata-package of his concern based on the filtering mask already set on filtering module on his set-top.

6.Complex textual metadata is interpreted in the textual metadata handling module. (e.g. It might be implemented metadata interpretor based on RDF/XML parser)

7.Selected UPI is used to UPI resolution.

-



-

6.Resolution module gets the locator(URL) for the content and Set-top captures it by the locator.

5.1.6.6.5 Program Index based navigation scenario

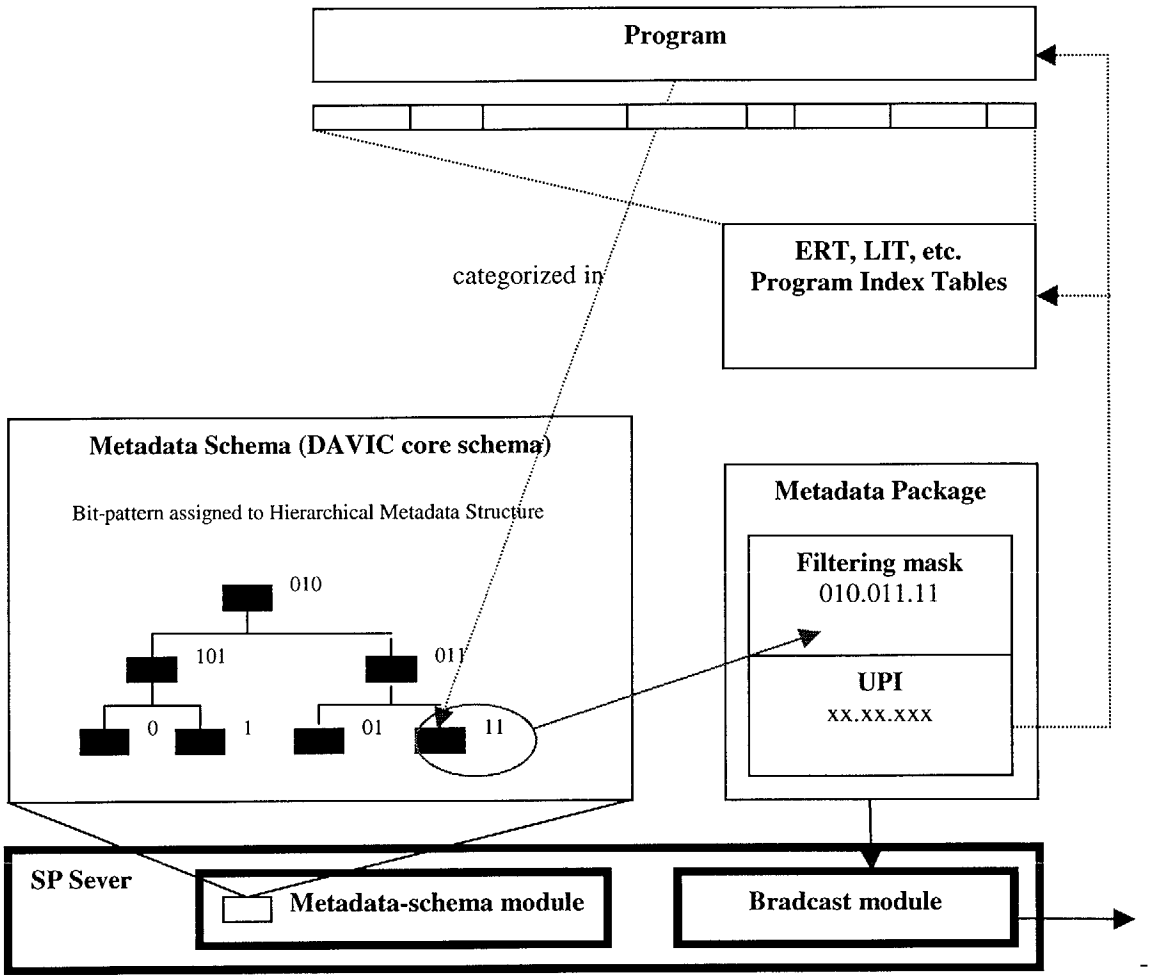
1.Service Provider (SP) creates the AV-program.

2.SP creates the ERT and LIT, other tables enables Program Index based navigation for the program.

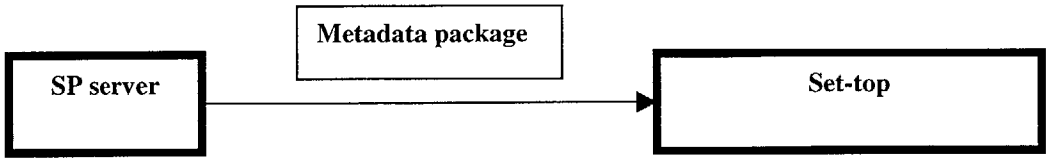
2.1.Selects the metadata category for the program based on the DAVIC core schema.

2.2.Encodes the filtering mask based on the bit-pattern syntax for DAVIC core schema.

2.3.Encodes the UPI for Program Index tables and the program.



3.SP broadcasts the matadata-package to the set-tops.



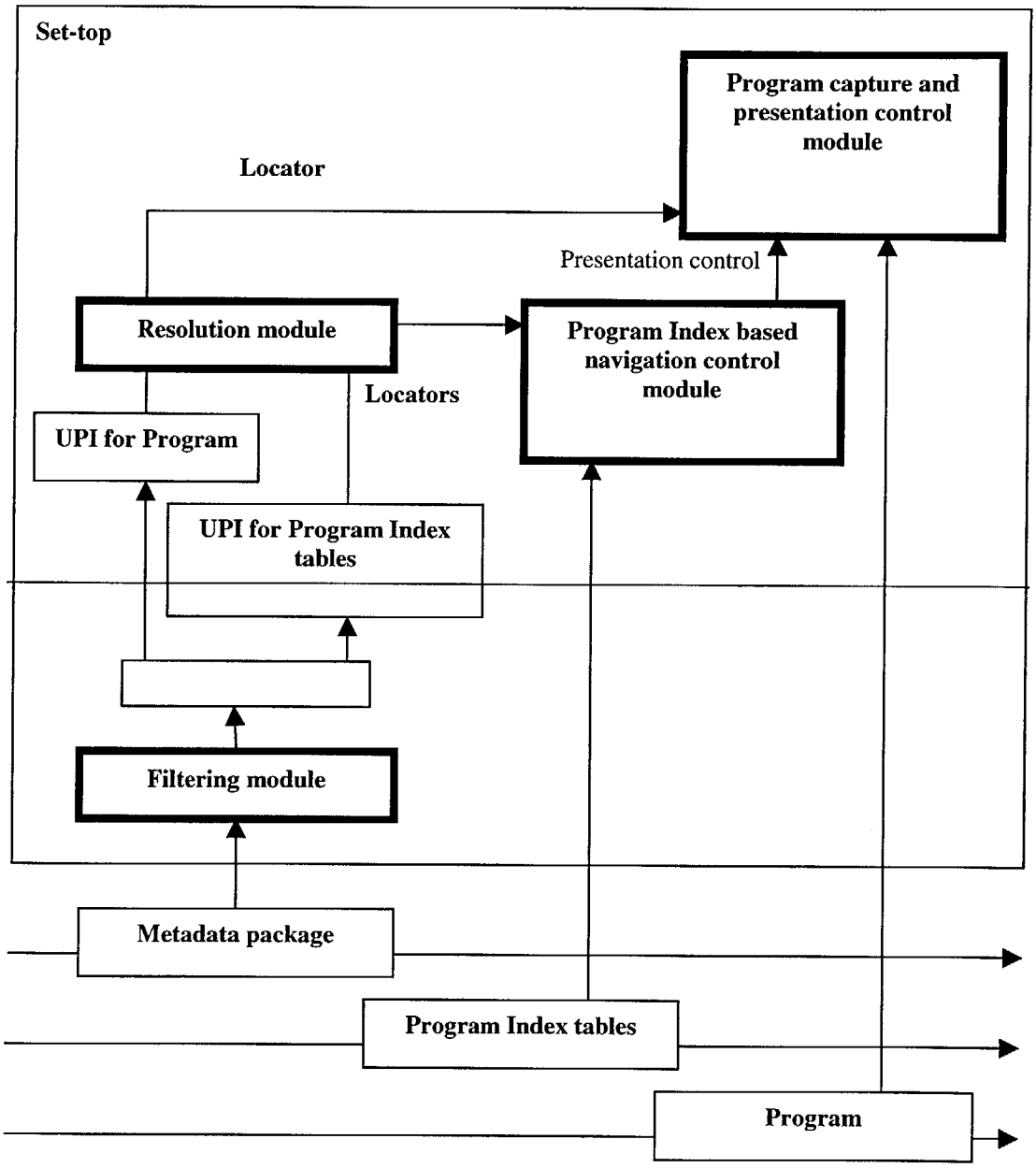
4.Set-top on Service Consumer (SC) extracts the matadata-package of his concern based on the filtering mask already set on filtering module on his set-top.

5.Set-top extracts the UPI for the Program Index tables and the program then asks UPI resolution module to resolve them.

6.Resolved locators for Program Index tables are used to capture those tables then captured tables are handled in Program Index based navigation control module.

7.Resolved locator for the program is used to capture the program then captured program is presented through program capture and presentation control module controled by Program Index based navigation control module.

-



5.2 Java APIs for TV Anytime and TV Anywhere - The "org.davic.storage" Package and Supporting Classes

This section principally describes extensions, in the form of additional Java APIs, to the application format defined by part 9 of DAVIC 1.4.1 specifications.

5.2.1 Objective

The objective of this API is to provide a mechanism for inter-operable applications to select content for recording, to monitor the status of those recordings and to playback content which has been recorded.

5.2.2 Requirements

This section is informative.

5.2.2.1 General Requirements

1. Support recording of a program including all required audio, video, data and any associated application where allowed by the recording device.
2. Must support individual ownership of recordings when used in a system supporting identification of multiple users.
3. Must be independent of particular recording devices as far as possible
4. Must support and be transparent between integrated recording devices and recording devices accessed via an in-home network (both IP and non-IP).
5. Must allow an underlying security framework to refuse access and permissions.
6. Support but not require interoperable applications only being able to access recordings made by that application or other applications from the same service provider.
7. Must be able to coexist with recordings programmed for a particular storage device other than through the API.

5.2.2.2 Recording Requirements

1. Specify something to be recorded based on a UPI.
2. Specify something to be recorded based on a UPI with extra preferences (e.g. must be recorded on a particular day)
3. Specify something to be recorded based on "time and channel".
4. Specify something to be recorded based on a URL.
5. Identify and express conflicts in planned recordings.
6. Obtain a list of planned recordings.
7. Query the status of a requested recording(s) (pending, failed)
8. Cancel a planned recording
9. Support but not require varying levels of access to recording capabilities by applications (none, user confirmation required, unlimited)
10. Differentiate between recordings made for the user and recordings made for the service provider. (different levels of user intervention may happen)

5.2.2.3 Playback Requirements

1. Support basic VCR functionality for playback control
2. Support DAVIC 1.4.1 playback control capabilities (e.g. language selection)

3. Where program segment data exists (see CC sub-group), support navigation between segments.

5.2.2.4 Storage Management Requirements

1. List stored recordings
2. Delete stored recordings
3. Support but not require quotas for service providers to allow them to store limited quantities of content whenever they wish (e.g. for content customisation)

5.2.3 General

The packages specified below model a storage system consisting of one or more lists of recordings to be made (class RecordingList) made up from objects of the RecordingListEntry class.

5.2.4 Specification of Supporting Classes

5.2.4.1 Extensions to the "org.davic.net" Package From DAVIC 1.4.1

5.2.4.1.1 org.davic.net.UPI

```
java.lang.Object
   |
   +----org.davic.net.UPI
```

public class **UPI**

extends Object

This class represents a uniform program identifier.

Constructors

```
public UPI(String string)
```

Construct a UPI from a string

Parameters:

string - the string describing the UPI

Methods

resolveLeaf

```
public Locator[] resolveLeaf() throws ReferenceException
```

Resolve a UPI which references a single piece of content to a set of locators for the actual piece of content. Each of the locators is equivalent to a repeat of the same piece of content as defined by the content provider.

Returns:

an array of locators for a piece of content including repeats (if any)

Throws: UnknownResolvingEntityException

if the ResolvingEntity specified in the UPI is not known

Throws: NoResolvingDataException

if there is no data available to resolve the UPI

Throws: MalformedReferenceException

if there is a syntax error in the UPI

Throws: NotLeafException

if the UPI does not reference a single piece of content

resolveSet

```
public UPI[] resolveSet() throws ReferenceException
```

Resolve a UPI which references a set of pieces of content into the UPIs for the individual items in the set. Each of the UPIs is equivalent to a different piece of content in a series or serial as defined by the content provider.

Returns:

an array of UPIs for the components of the set

Throws: UnknownResolvingEntityException

if the ResolvingEntity specified in the UPI is not known

Throws: NoResolvingDataException

if there is no data available to resolve the UPI

Throws: MalformedReferenceException

if there is a syntax error in the UPI

Throws: NotSetException

if the UPI does not reference a single piece of content

5.2.4.1.2 ReferenceException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----org.davic.net.ReferenceException
```

public abstract class **ReferenceException**

extends Exception

Base class for exceptions related to content references in the DAVIC net package

Constructors

```
public ReferenceException()
```

Constructs a ReferenceException with no detail message

```
public ReferenceException(String detail)
```

Constructs a ReferenceException with the specified detail message.

Parameters:

detail - the detail message

5.2.4.1.3 MalformedReferenceException

```
java.lang.Object
   |
   +----java.lang.Throwable
            |
            +----java.lang.Exception
                     |
                     +----org.davic.net.ReferenceException
                              |
                              +----org.davic.net.MalformedReferenceException
```

public class **MalformedReferenceException**

extends ReferenceException

Exception generated in case of syntax errors in UPIs

Constructors

```
public MalformedReferenceException()
```

Constructs a MalformedReferenceException with no detail message

```
public MalformedReferenceException(String detail)
```

Constructs a MalformedReferenceException with the specified detail message.

Parameters:

detail - the detail message

5.2.4.1.4 NonResolvableReferenceException

```
java.lang.Object
   |
   +----java.lang.Throwable
            |
            +----java.lang.Exception
                     |
                     +----org.davic.net.ReferenceException
                              |
                              +----
org.davic.net.NonResolvableReferenceException
```

public class **NonResolvableReferenceException**

extends ReferenceException

Exception generated when a reference must be resolvable and is not at the time in question. This exception is only generated for methods where a non-resolvable reference is a fatal error. For methods where a non-resolvable reference isn't a fatal error, this exception will not be generated.

Constructors

```
public NonResolvableReferenceException()
```

Constructs a NonResolvableReferenceException with no detail message

```
public NonResolvableReferenceException(String detail)
```

Constructs a NonResolvableReferenceException with the specified detail message.

Parameters:

detail - the detail message

5.2.4.1.5 NoResolvingDataException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----org.davic.net.ReferenceException
                           |
                           +----
org.davic.net.NonResolvableReferenceException
                                   |
                                   +----
org.davic.net.NoResolvingDataException
```

public class **NoResolvingDataException**

extends NonResolvableReferenceException

Exception generated when there is no data currently available to resolve a reference. This exception may indicate a temporary condition. Calling the same method a short time earlier or later may not yield an exception if the environment changes.

Constructors

```
public NoResolvingDataException()
```

Constructs a NoResolvingDataException with no detail message

```
public NoResolvingDataException(String detail)
```

Constructs a NoResolvingDataException with the specified detail message.

Parameters:

detail - the detail message

5.2.4.1.6 NotLeafException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----org.davic.net.ReferenceException
                           |
                           +----org.davic.net.NotLeafException
```

public class **NotLeafException**

extends ReferenceException

Exception generated when a UPI should be a reference to a single piece of content and is not.

Constructors

`public NotLeafException()`

Constructs a NotLeafException with no detail message

`public NotLeafException(String detail)`

Constructs a NotLeafException with the specified detail message.

Parameters:

detail - the detail message

5.2.4.1.7 NotSetException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----org.davic.net.ReferenceException
                           |
                           +----org.davic.net.NotSetException
```

public class **NotSetException**

extends ReferenceException

Exception generated when a reference should be a reference to a set of pieces of content (e.g. a series or serial) and is not.

Constructors

`public NotSetException()`

Constructs a NotSetException with no detail message

`public NotSetException(String detail)`

Constructs a NotSetException with the specified detail message.

Parameters:

detail - the detail message

5.2.4.1.8 UnknownResolvingEntityException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----org.davic.net.ReferenceException
                           |
                           +----
```

```
org.davic.net.NonResolvableReferenceException
                                        |
                                        +----
org.davic.net.UnknownResolvingEntityException
```

public class **UnknownResolvingEntityException**

extends NonResolvableReferenceException

Exception generated when a UPI is not resolvable because the environment does not know about the resolving entity for the UPI.

Constructors

`public UnknownResolvingEntityException()`

Constructs a UnknownResolvingEntityException with no detail message

`public UnknownResolvingEntityException(String detail)`

Constructs a UnknownResolvingEntityException with the specified detail message.

Parameters:

detail - the detail message

5.2.5 Media Playback API Specification

The Java Media Framework (JMF) already selected for DAVIC 1.4.1 will be used for this purpose.

5.2.6 Recording API Specification

5.2.6.1 Exceptions

5.2.6.1.1 org.davic.storage.recording.RecordingException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----org.davic.storage.recording.RecordingException
```

public abstract class **RecordingException**

extends Exception

Base class for exceptions in the org.davic.storage.recording package

Constructors

`public RecordingException()`

Constructs a RecordingException with no detail message

`public RecordingException(String detail)`

Constructs a RecordingException with the specified detail message.

Parameters:

detail - the detail message

5.2.6.1.2 org.davic.storage.recording.PermissionDeniedException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----org.davic.storage.recording.RecordingException
                           |
                           +----
org.davic.storage.recording.PermissionDeniedException
```

public class **PermissionDeniedException**

extends RecordingException

This exception is thrown when an application is refused access to a particular piece of information or function. Examples of this can include :-

- attempts by applications from one service provider to access recording list entries setup by applications from another service provider.
- on a system supporting identification of multiple users, attempts to access recording list entries from other users.
- deleting a recording.

Constructors

`public PermissionDeniedException()`

Constructs a PermissionDeniedException with no detail message

`public PermissionDeniedException(String detail)`

Constructs a PermissionDeniedException with the specified detail message.

Parameters:

detail - the detail message

5.2.6.1.3 org.davic.storage.recording.QuotaFullException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----org.davic.storage.recording.RecordingException
                           |
                           +----
org.davic.storage.recording.QuotaFullException
```

public class **QuotaFullException**

extends RecordingException

This exception is thrown for application or service provider oriented recordings where a quota system exists and the quota for the application or service provider does not allow the recording.

Constructors

`public QuotaFullException()`

Constructs a QuotaFullException with no detail message

`public QuotaFullException(String detail)`

Constructs a QuotaFullException with the specified detail message.

Parameters:

detail - the detail message

5.2.6.1.4 org.davic.storage.recording.InappropriateContentException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----org.davic.storage.recording.RecordingException
                           |
                           +----
org.davic.storage.recording.InappropriateContentException
```

public class **InappropriateContentException**

extends RecordingException

Exception generated when a UPI or Locator references content which is inappropriate for the recording operation requested.

Constructors

`public InappropriateContentException()`

Constructs a InappropriateContentException with no detail message

`public InappropriateContentException(String detail)`

Constructs a InappropriateContentException with the specified detail message.

Parameters:

detail - the detail message

5.2.6.2 Interfaces

5.2.6.2.1 RecordingListListener

public interface **RecordingListListener**

Objects wishing to be notified about changes in state of recording list entries must implement this interface.

Methods

receiveRecordingListEvent

```
public abstract void receiveRecordingListEvent(RecordingListEvent e)
```

This method is called to send an event to the listener.

Parameters:

e - the event to be sent

5.2.6.3 Classes

5.2.6.3.1 org.davic.storage.recording.RecordingList

```
java.lang.Object
   |
   +----org.davic.storage.recording.RecordingList
```

public class **RecordingList**

extends Object

This class represents a list of recordings to make.

Methods

getUserInstance

```
public static RecordingList getUserInstance()
```

Get an instance to be used for user oriented recordings. On systems supporting identification of multiple users, there may be an instance of this class for each user. On systems not supporting identification of multiple users, all calls to this method will probably return the same object.

The implementation of the API may request user confirmation of recordings put into this list.

Returns:

a RecordingList for recordings potentially associated with a user

getApplicationInstance

```
public static RecordingList getApplicationInstance()
```

Get an instance to be used for application or service provider oriented recordings. The recording list returned will be associated with the application or source of the application (e.g. service provider).

If there is a quota mechanism for service providers then recordings made in this list will be allocated to that quota. Less (or no) user confirmation may be required than for recordings made in the user oriented recording list.

Receivers are not required to support application or service provider oriented recording.

Returns:

a RecordingList for recordings potentially associated with a service provider or null if application / service provider oriented recording is not supported.

Record

```
public RecordingListEntry Record(UPI upi,
                                 String title) throws RecordingException,
ReferenceException
```

Setup a recording based on a UPI. Recordings which cannot be made because of insufficent available space or resource conflicts will not generate any error report as these situations can change dynamically based on user intervention. The getStatus method on recording list entry should be used to identify these situations.

Parameters:

upi - the UPI for the item to be recorded

title - the title to use for the item to be recorded

Returns:

a reference to the RecordingListEntry for the item.

Throws: QuotaFullException

thrown for application or service provider oriented recordings when used with a quota system and the quota does not allow the recording.

Throws: MalformedReferenceException

thrown when the UPI provided contains a syntax or similar formatting error.

Throws: UnknownResolvingEntityException

thrown when the UPI provided contains a resolving entity which is not known

Throws: InappropriateContentException

thrown when the the UPI provided is not appropriate for something to be recorded.

Record

```
public RecordingListEntry Record(UPI upi,
                                 String title,
                                 Locator locator) throws
RecordingException, ReferenceException
```

Setup a recording based on a UPI with a suggestion of how the UPI might be resolved to a location. Recordings which cannot be made because of insufficent available space or resource conflicts will not generate any error report as these situations can change dynamically based on user intervention. The getStatus method on recording list entry should be used to identify these situations.

Parameters:

upi - the UPI for the item to be recorded

title - the title to use for the item to be recorded

locator - a suggestion for which instance of the program to record for cases where there is a choice (e.g. the program is repeated)

Returns:

a reference to the RecordingListEntry for the item.

Throws: QuotaFullException

thrown for application or service provider oriented recordings when used with a quota system and the quota does not allow the recording.

Throws: MalformedReferenceException

thrown when the UPI or locator provided contains a syntax or similar formatting error.

Throws: UnknownResolvingEntityException

thrown when the UPI provided contains a resolving entity which is not known

Throws: InappropriateContentException

thrown when the the UPI or locator provided is not appropriate for something to be recorded.

Record

```
public RecordingListEntry Record(Locator locator,
                                 String title) throws RecordingException,
ReferenceException
```

Setup a recording for a specific location. Recordings which cannot be made because of insufficent available space or resource conflicts will not generate any error report as these situations can change dynamically based on user intervention. The getStatus method on recording list entry should be used to identify these situations.

Parameters:

locator - the locator for the item to record

title - the title to use for the item to be recorded

Returns:

a reference to the RecordingListEntry for the item.

Throws: QuotaFullException

thrown for application or service provider oriented recordings when used with a quota system and the quota does not allow the recording.

Throws: InappropriateContentException

thrown when the the locator provided is not appropriate for something to be recorded.

Throws: MalformedReferenceException

thrown when the locator provided contains a syntax or similar formatting error.

getEntries

```
public RecordingListEntry[] getEntries()
```

Return the planned recordings in this list.

Returns:

an array containing all the entries in this recording list.

getRecordings

```
public Recording[] getRecordings()
```

Return the stored recordings made from this list. This method need only return those which are available for playback at the time the method is called.

Returns:

an array of recordings

addRecordingListListener

```
public void addRecordingListListener(RecordingListListener l)
```

Add an event listener for changes in status of recording list entries.

Parameters:

l - the listener to be registered

removeRecordingListListener

```
public void removeRecordingListListener(RecordingListListener l)
```

Remove a registed event listener for changes in status of recording list entries. If the listener specified is not registered then this method has no effect.

Parameters:

l - the listener to be removed.

5.2.6.3.2 org.davic.storage.recording.RecordingListEntry

```
java.lang.Object
   |
   +----org.davic.storage.recording.RecordingListEntry
```

public class **RecordingListEntry**

extends Object

This class represents one entry in a list of recordings to be made.

The implementation may restrict access to the information contained in this entry depending on an underlying security model.

- For user oriented recordings, on systems supporting identification of multiple users, access should be refused to recording list entries from users other than the current one.
- For application or service provider oriented recordings, access should be refused to recording list entries from other applications or service providers.

Variables

PENDING

```
public static final int PENDING
```

Indication that a recording is still pending and is possible to make as far as is known at this time.

FAILED

```
public static final int FAILED
```

Indication that a recording failed.

IMPOSSIBLE

```
public static final int IMPOSSIBLE
```

Indication that a recording is impossible to make as far as is known at this time.

UNKNOWN

```
public static final int UNKNOWN
```

Indication that the status of a recording is unknown. One example of this would be recordings where the size and time are not yet known.

Methods

getLocator

```
public Locator getLocator() throws PermissionDeniedException
```

return the Locator specifying the recording to be made for a recording list entry.

Returns:

the Locator specifying the recording to be made for a recording list entry. If the entry was generated from a UPI and that UPI has not yet been resolved then null will be returned.

Throws: PermissionDeniedException

if the application is not authorised to access the entry.

getConflicts

```
public RecordingListEntry[] getConflicts() throws
PermissionDeniedException
```

check whether there are any conflicts with this recording list entry

Returns:

an array containing any other RecordingListEntry objects which conflict with this one. If there are no conflicts then null will be returned.

Throws: PermissionDeniedException

if the application is not authorised to access the entry.

cancel

```
public void cancel() throws PermissionDeniedException
```

cancel a recording list entry.

Throws: PermissionDeniedException

if the application is not authorised to access the entry.

getCreationDate

```
public Date getCreationDate() throws PermissionDeniedException
```

get the date when a recording list entry was created

Returns:

the date when a recording list entry was added to the list

Throws: PermissionDeniedException

if the application is not authorised to access the entry.

5.2.6.3.3 org.davic.storage.recording.Recording

```
java.lang.Object
    |
    +----org.davic.storage.recording.Recording
```

public class **Recording**

extends Object

This class represents a completed recording.

Methods

getCreationDate

```
public Date getCreationDate()
```

Get the date when the recording was made

Returns:

the date when the recording was made

getDuration

```
public long getDuration()
```

Get the duration of the recording in time

Returns:

the duration of the recording in seconds

getTitle

```
public String getTitle()
```

Get the title of the recording

Returns:

the title of the recording provided when the recording was setup

getLocator

```
public Locator getLocator()
```

Get a locator which can be used to play back the recording.

Returns:

a locator which can be used to play back the recording.

delete

```
public void delete() throws PermissionDeniedException
```

Delete a recording.

Throws: PermissionDeniedException

if deleting the recording is not allowed.

5.2.6.3.4 RecordingListEvent

```
java.lang.Object
   |
   +----org.davic.storage.recording.RecordingListEvent
```

public class **RecordingListEvent**

extends Object

This class is used when the status of entries in a recording list changes. These events are only generated when the status of entries changes while there is a registered listener for these events. Changes generated while there is not a registered listener will not be reported.

Constructors

```
public RecordingListEvent(RecordingListEntry changed[],
                          Object RecordingList)
```

Constructor for the event.

Methods

getChangedEntries

```
public RecordingListEntry[] getChangedEntries()
```

Get the set of recording list entries whose status has changed.

Returns:

an array of recording list entries whose status has changed.

getSource

```
public Object getSource()
```

Get the RecordingList which was the source of this event

Annex A

Example of Metadata Schema

Category	Sub-category 1	Sub-category 2	Sub-category 3	Sub-category 4	Sub-category 5
Type (DC)	Programme Group (DC – collection)	Group size			
		Group type	Unspecified		
			No relation		
			Serial		
			Season		
			Mini-serie		
			Special		
			Other		
		Other			
	Programme (DC – event)	Programme type	Single		
			Episode/part		
			Other		
		Audio programme	Full		
			Segment	Segment type	
				Other	
			Object		
			Other		
		Video programme	Full		
			Segment	Segment type	
				Other	
			Object		
			Other		
		A/V programme	Full		
			Segment		
			Object		
			Other		
		Other			
	Application (DC)	Full			
		Routine			
		Other			

	Data (DC)	Stream			
		Stream element			
		Other			
	Other				
Title (DC)	Title				
	Original title				
	Subtitle				
	Number in a series				
	Other				
Subject, keywords (DC)	Category/Genre	Unspecified			
		music	Music Videos / The Charts		
			Pop music		
			Rock		
			World Music		
			Jazz		
			Folk		
			Classical	Serious	Symphonic
					Chamber
					Opera
					Other
				Light	
				Other	
			Religious		
			Experimental		
			Contemporary		
			Other		
		arts	Performing art	Ballet	
				Dance	
				Theatre	
				Other	
			Culture	Literature	
				Poetry	
				Other	
			Fine Arts		
			Experimental Arts		
			Traditional Arts		

	Popular arts	
	Other	
Entertainment	Shows	Talk shows
		Game shows
		Quiz shows
		Variety shows
		Talent shows
		Awards shows
		Other
	Live events	
	Magazine	
	Reportage	
	Other	
Science & Technology	Technology	Computer/internet
		Space
		Automation
		New technology
		Other
	Science	Natural Sciences
		Physical Sciences
		Medical Sciences
		Physiology
		Psychology/sociology
		Other
	Other	
Nature & Natural History	Nature	
	Natural History	
	The Environment	
	Other	
News & Current Affairs	Current Affairs	
	Financial / business News	International News
		Regional News
		Local News
		News Flash
		News Summary

		Stock market
		Magazine
		Other
	News	International News
		Regional News
		Local News
		News Flash
		News Summary
		Magazine
		Reportage
		Other
	Politics	Parliamentary coverage
		Political Debate
		Party political broadcast
		Ministerial/Presidential broadcast
		Other
	Weather	International
		Regional
		Local
		Alert Flash
		Special report
		Other
	Traffic/transport	Flash
		Special report
		Other
	Other	
Education	Education Issues	Pre-School
		Primary Education
		Secondary Education
		Higher Education
		Other
	Educational Programmes	Schools Programmes
		Open University
		Course
		Instruction
		Other
	Other	

Religion	Religious Service	
	Spirituality	
	Teology	
	Other	
People & Life style	Life style	
	Ethnic	
	Gay/Lesbian	
	Special needs	
	Women's	
	Men's	
	Elderly	
	Other	
Hobby, Leisure & Interest	Consumer affairs	
	Leisure time	Auto & motor
		Gym & Fitness
		Fashion
		Shopping
		Other
	Holiday & Travel	Tourism
		Travel Report
		Foreign Countries
		Other
	Hobby	DIY
		Cooking
		Handicraft
		Home Interior
		Motoring
		Gardening
		Other
	Other	
Children's, Youth & Teenage	Children's	Cartoon
		Comic
		Puppets
		Other
	Pre-School	
	Teenagers	

	Youth
	Informational/educational/school programmes
	Other
Film & Drama	Historical
	Classic
	Melodrama
	Soap
	Biographical
	Crime
	Thriller
	Spy
	Hospital
	Action
	Adventure
	Science fiction
	Horror
	War
	Western
	Mystery
	Law
	Detective
	Police
	Fantasy
	Eroticism
	Adult
	Cult
	Classic
	Romance
	Musical
	Cartoon
	Court-metrage
	Other
Comedy	Situation Comedy
	Black Comedy
	Farce
	Satire
	Parody
	Impressionists' shows

	Other	
History	Prehistory/Archaeology	
	Ancient	
	Medieval	
	Modern	
	Contemporary	
	Other	
Social, Public & Community	Appeal Programmes	
	Crime-fighting programmes	
	Public information	
	Other	
Sports	Sports (general)	
	Sport events	Olympic games
		world cup
		Other
	Sport coverage	Sports magazines
		American football
		Athletics
		Baseball
		Basketball
		Boxing
		Combat Sport
		Cricket
		Cycling
		Darts
		Equestrian
		Football (soccer)
		Hockey
		Martial sports
		Motor Sports
		Rugby
		Squash
		Team Sports
		Tennis
		Volley ball
		Water Sports
		Winter Sports

				Team sports
				Other
			Other	
		Advertising	Trailer	
			Announcement	
			Other	
		Other		
	Other			
Description (DC)	Description			
	Short description, synopsis, summary / plot			
	Summary of past plot			
	Postview			
	Script	Title		
		Main events		
		Location		
		Main objects		
		Dialogues		
		Other		
	Visual feature	Focus type		
		Camera operation		
		Activity measure		
		Contrast		
		Luminosity		
		Colour		
		Other		
	Audio featires	Type of content		
		Energy		
		Pitch		
		Speech		
		Other		
	Object type	Shape		
		Colour		
		Texture		
		Mouvement		
		Location		
		OCI information		
		Other		
	Review	Source		

		Rating	
		Max Rating	
		Text values	
		Comments	
		Other	
	Award	Award name	
		Date	
		Reason	
		Other	
	Casting	Actor_name	Gender
			Character
			Other
		Other	
	Role_description		
	Author_name		
	Producer_name		
	Director_name		
	Photo_director_name		
	Rating	Categories a,b,c,d,	
		Other	
	Other		
Source (DC)	*Void*		
	Other		
Coverage (DC)	*Void*		
	Other		
Creator (DC)	Author		
	Director		
	Producer		
	Other		
Publisher (DC)	Publisher		
	Provider		
	Other		
Contributor (DC)	Contributor		
	Other		
Rights	Access conditions	Unrestricted	

(DC)				
		Licensee		
		Distribution coverage	International	
			National	
			Regional	
			Other	
		Usage restrictions	Unrestricted	
			Editing restrictions	
			Copy restriction	Unlimited
				Number of copies
				Other
			Other	
		Other		
	Unique Identifier			
	Country of production			
	Production date			
	Other			
Date, time	Date of creation			
(DC)				
	Scheduled time			
	Delivery starting time			
	Delivery end time			
	Other			
Language	Original			
(DC)				
	Audio			
	Subtitle			
	Other			
Format	Audio			
(DC)				
		Compression system		
		Single mono channel		
		Dual mono channel		
		Stereo (2 channel)		
		Multi-channel, multi-lingual		
		Multi-channel, surround sound		

		Reserved for future use		
		Audio description for the visually impaired		
		Audio for the hard of hearing		
		Scalable sound		
		Other		
	Video	Reserved for future use		
		Compression type		
		Horizontal size		
		Vertical size		
		Aspect ratio	4:3 aspect ratio	
			16:9 aspect ratio with pan vectors	
			16:9 aspect ratio without pan vectors	
			> 16:9 aspect ratio	
			Other	
		Other		
	Data	Data System		
		Data format		
		Other		
	Platform	File/content storage size		
		Bit rate	CBR	
			VBR	Average rate
				Maximum rate
				Minimum rate
				Other
			Other	
		Other		
	Other			
Identifier (DC)	UPI	ContID		
		RE_ID		
		RE_Address		
		Links	Multimedia links	
			UPIs	
			Back channel	
			Other	
		Other		
	URL	Service nane	Services description	

	Other	
Channel name		
Bouquet name	Bouquet association	
Network name	Network information	
	Other	
Service_descriptio n		
Delivery system	Terrestrial	
	Satellite	"
	Cable	"
	ATM	"
	Other	
Service Type	Digital television service	
	digital radio sound service	
	Teletext service	
	NVOD reference service	
	NVOD time-shifted service	
	Mosaic service	
	PAL coded signal	
	SECAM coded signal	
	D/D2-MAC	
	FM Radio	
	NTSC coded signal	
	~~D~~data broadcast service	
	Other	
Version		
Elements		
Component_descri ptor		
Origination_date_a nd_time		
Time_and_date		
Next_time		
Actual/relative_tim e		
Bandwidth_needed		
Caption_link		

Time_shifted_even
t_descriptor

Telephone_descrip
tor

Other

Other

Annex B

Example of Metadata Instance of DAVIC Core Schema

```
/*
        Example
*/

formatID: "#x0001"

nodeName: "Subject_keywords(DC)/Category_Genre/music/Pop_music";

nodeName: "Creator(DC)/Author";
attributeName: "FirstName";
attributeValue: "Super";
attributeName: "LastName";
attributeValue: "Man";
```

Annex C

Example of Metadata Schema for DAVIC Core Schema

```
/*
        Example
*/

formatID: "#x0002";
schemaID: "#x0001";

nodeName: "Subject_keywords(DC)";
offset: "#0";
bitPattern: "010";
nodeName: "Description";
offset: "#0";
bitPattern: "011";
/* .... */
nodeName: "Subject_keywords(DC)/Category_Genre";
offset: "#3";
bitPattern: "010";
nodeName: "Subject_keywords(DC)/Category_Genre/music";
offset: "#6";
bitPattern: "00001";
nodeName: "Subject_keywords(DC)/Category_Genre/arts";
offset: "#6";
bitPattern: "00010";
/* .... */
nodeName: "Subject_keywords(DC)/Category_Genre/music/Pop_music";
offset: "#11";
bitPattern: "0001";
nodeName: "Subject_keywords(DC)/Category_Genre/music/Rock";
offset: "#11";
bitPattern: "0010";
/* .... */
```

Annex D

Example of Metadata Schema Extension of DAVIC Core Schema

```
/*
        Example
*/

formatID: "#x0002";

schemaID: "#x0002";
targetSchemaID: "#x0001";

nodeName: "Subject_keywords(DC)/Category_Genre/music/Rock/UK";
offset: "#15";
bitPattern: "01";

nodeName: "Subject_keywords(DC)/Category_Genre/music/Rock/Japan";
offset: "#15";
bitPattern: "10";

nodeName: "Subject_keywords(DC)/Category_Genre/music/Rock/USA";
offset: "#15";
bitPattern: "11";

/* .... */
```